UNIVERSITÄT LEIPZIG

ASV
Automatische Sprachverarbeitung

Universität Leipzig - Institut für Informatik

Seminararbeit

zum Thema

Übersicht über Textformate

Autor: Hendrik Sawade

Veröffentlichungsdatum: Leipzig, den 20. September 2017

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Auflistungen

1 Einleitung

Das Ziel dieser Arbeit ist es, eine grobe Übersicht verschiedener Textdatenformate zu geben. Dabei wird zwischen Block- und Markup-Formate unterschieden, welche zu Beginn erklärt werden. Für die hier vorgestellten Formate werden zunächst deren Bedeutung und Funktionsweise beschrieben. Im Anschluss werden die Blockformate Microsoft Doc 2.1.1 und Portable Document Format (PDF) 3.1.3 erläutert, welche Dokumenten-Formate darstellen. Danach werden einige Markup-Formate aufgezeigt, welche sich in Offline-Formate 3.1, Dokumenten 3.2 - und Online-Formate 3.3 unterteilen. Beispiele hierfür sind EPUB, Latex und Markdown. Außerdem werden zu jedem Format Gründe für die Entwicklung, Zielgruppe, Aufbau sowie deren Syntax in Form von Beispielen erläutert. Ebenso werden Vor- und Nachteile zu den jeweiligen Formaten genannt. Abschließend folgt eine Zusammenfassung der Formate in Form einer tabellarischen Übersicht in Abschnitt 4, in welcher Gemeinsamkeiten und Unterschiede, sowie Alleinstellungsmerkmale in Abschnitt 5 aufgeführt werden. Hierbei muss berücksichtigt werden, dass die Tabellen in Abschnitt 4 eine Selbsteinschätzung des Autors darstellen.

1.1 Block-Formate

Dokumenten-Formate, welche in die Block-Struktur fallen, dienen der Gliederung und Formatierung von Texten und anderen Daten, sowie der Gestaltung des Inhalts.

Über bestimmte Auszeichnungen wird festgelegt, wie der Text gestaltet wird oder dieser bestimmte Formatierungen besitzen soll. Die Auszeichnungen stehen dabei meistens am Anfang und Ende des jeweiligen Dokuments. Strukturen können aber auch innerhalb der Auszeichnungen stehen, sodass die Anwendung durch diese erkennt, wie der Text dargestellt und formatiert werden soll. Die einzelnen Abschnitte erhalten dabei im Text genaue Positionsangaben für einzelne Auszeichnungen, wie es beispielsweise bei Microsoft Doc (Abschnitt 2.1.1) gehandhabt wird [1, 2, 3].

1.2 Markup-Formate

Dokumenten-Formate, welche sich in der Markup-Struktur-Familie befinden, dienen ebenso wie die Block-Struktur der Gliederung und Formatierung von Texten und anderen Daten sowie der Gestaltung deren Inhalten. Der Unterschied besteht in dem Aufbau der Formate. Mit Markup-Struktur-Formaten werden Eigenschaften, Zugehörigkeiten und Darstellungsformen von Abschnitten eines Textes (Zeichen, Wörter, Absätze usw.) oder einer Datenmenge beschrieben. Dies erfolgt in den meisten Fällen durch das Ausstatten mit Tags. Dadurch wird eine klare Trennung der Struktur und Darstellung erreicht. Einige dieser Tags dienen der Struktur, andere Tags sind für die Darstellung des Inhaltes zuständig. Beispielsweise strukturiert HTML (Abschnitt 3.3.2) den Text und CSS gestaltet ihn [4, 5].

2 Block-Formate

2.1 Dokumenten-Formate

Die Formate Doc und PDF dienen der Erstellung von Dokumenten, welche Texte, Bilder, Tabellen etc. beinhalten können. Damit sind Dokumente wie Bürodokumente einfacher zu handhaben.

2.1.1 Microsoft Doc

Das Format Doc wurde von Microsoft bis Office 2003 verwendet und ist Block-orientiert. Es diente zur Dokumentenerstellung und zum Datenaustausch von Bürodokumenten. Sämtliche Zeichendaten, welche das Dokument beinhaltet, befinden sich im Word Document-Datenstrom. Am Anfang dieses Datenstroms befindet sich ein File Information Block (FIB). Die genannte Struktur verweist auf alle Daten, welche in der Datei enthalten sind. In diesen Daten befinden sich die eigentlichen Dokumentdaten, wie zum Beispiel die Texte oder Bilder. Die Speicherung des Stroms erfolgt binär und ist damit nur noch für spezielle Programme wie Word lesbar. Außerdem ist Doc kein freies Format und darf nicht ohne Weiteres in anderen Anwendungen verwendet werden. Das Format wurde schließlich ab Office 2007 durch das Microsoft Office Open XML (Abschnitt 3.2.2) abgelöst, weil Microsoft einen offenen Standard schaffen wollte. Aus diesem Grund hat heute Doc kaum noch eine Bedeutung und findet selten Verwendung in der Dokumentenerstellung. Deshalb wird Doc in dieser Arbeit nicht weiter betrachtet [3].

2.1.2 Portable Document Format

Geschichte / Grund der Entwicklung

PDF ist ein plattformunabhängiges Dateiformat für Dokumente, welches 1993 von der Firma Adobe Systems entwickelt und als offener Standard bereit gestellt wurde. Seitdem entwickelt Adobe das Format aktiv weiter und es steht momentan in Version 2.0 zur Verfügung.

Ziel war es, ein Dateiformat zu entwickeln, welches ein Dokument vom ursprünglichen Anwendungsprogramm, Betriebssystem oder Hardware originalgetreu darstellt. Das PDF wird dabei aus dem Originalformat erzeugt und kann anschließend in einem PDF-Reader betrachtet werden. Die Norm wurde schließlich mit der ISO 15930 für PDF/X festgelegt. Somit ist das PDF-Format standardisiert und auf sämtlichen Plattformen verwendbar.

Technisches

Jede PDF-Datei beinhaltet ein Dokument, welches eine vollständige Beschreibung des Seiteninhalts mit einem fixen Layout enthält. Im Layout befinden sich Texte, Schriften, Abbildungen und weitere Informationen, die zum Anschauen des Dokuments erforderlich sind. Eine solche Datei entspricht immer genau einem Dokument. Außerdem können in diesem Dokument eine beliebige Anzahl weiterer binäre Dateien in unterschiedlichen Formaten sowie andere PDF-Dateien eingebettet sein. Eine nachträgliche Bearbeitung der PDF-Datei ist nicht vorgesehen und muss daher in der Ursprungs-Datei vorgenommen werden.

Das PDF-Format ist eine vektorbasierte Seitenbeschreibungssprache, welche die Skalierbarkeit der Darstellung erlaubt. Der wesentliche Unterschied zwischen dem PDF-Standard und anderen Beschreibungs- und Auszeichnungssprachen wie SGML oder HTML besteht in der Layouttreue. Das PDF-Format beschreibt ein Layout, welches zuvor von einem anderen Erstellungsprogramm erzeugt wurde, in einer vom Drucker und von Voreinstellungen unabhängigen originalgetreuen Form.

Seit der Entwicklung von PDF wurden mehrere Versionen des PDF-Standards veröffentlicht, welche für unterschiedliche Einsatzzwecke entwickelt wurden. So gibt es beispielsweise PDF/X für die Druckvorstufe, PDF/E für CAD-Dateien im 2D- und 3D-Format, PDF/UA für barrierefreie Dokumente, PDF/VT für Datendruck mit variablen Inhalten und PDF/A für die Langzeitarchivierung von Dokumenten.

PDF/A beinhaltet Einschränkungen, da es für die rechtssichere elektronische Langzeitarchivierung von Dokumenten geeignet sein soll. Das einfache PDF-Format garantiert keine (identische und unveränderte) Reproduzierbarkeit in unterschiedlichen PDF-Viewern über einen langen Zeitraum oder die vollständige Unabhängigkeit von einer Software und dem Ausgabegerät. Um diese Funktionen aber gewährleisten zu können, müssen bestimmte Eigenschaften der PDF-Spezifikation genau definiert und eingeschränkt werden. So ist für die PDF/A-Konformität unter anderem erforderlich, dass alle Schriftarten in das Dokument eingebettet werden. Die Verschlüsselung des Dokuments, transparente Grafiken und die Wiedergabe von Audio- und Video-Inhalten sind nicht erlaubt [6]. Des Weiteren existiert PDF/A in drei Versionen, bzw. Konformitätsstufen: PDF/A-1, PDF/A-2 und PDF/A-3. Die Unterschiede ergeben sich aus diversen Anforderungen [7]:

- Level 1 conformance (PDF/A-1) – Sowohl eindeutige visuelle Reproduzierbarkeit, als auch Abbildbarkeit von Text nach Unicode und inhaltliche Strukturierung des Dokuments
- Level 2 conformance (PDF/A-2) – Lediglich eindeutige visuelle Reproduzierbarkeit
- Level 3 conformance (PDF/A-3) – Entspricht Level 2 plus Abbildbarkeit des Textes in Unicode-Zeichen

Aufbau

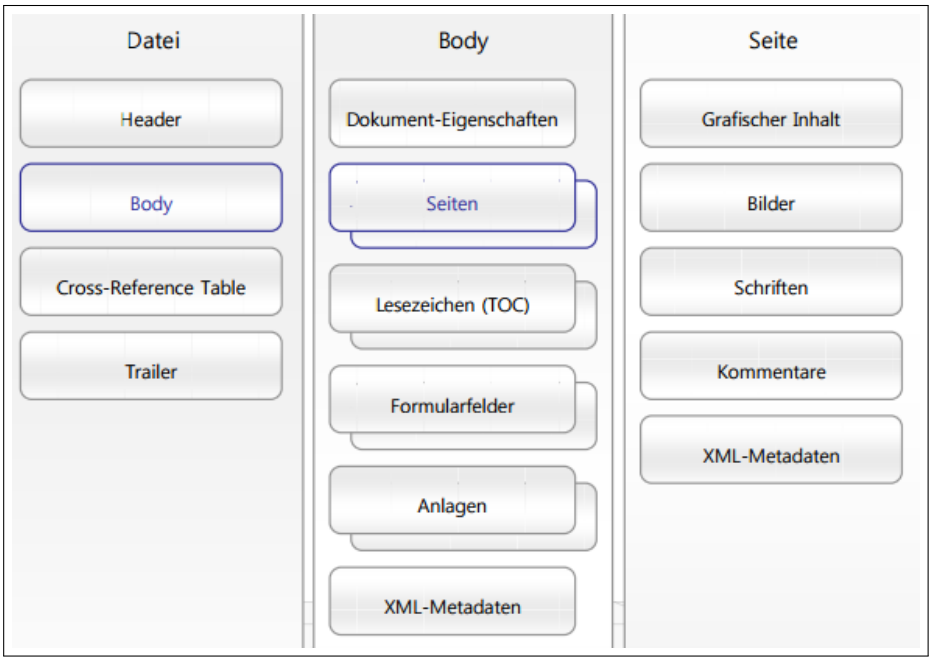

Abbildung 1: Cross-Reference Table [6]

Die Struktur einer PDF-Datei kann dabei wie folgt dargestellt werden [6]:

- Header – Identifiziert Version der PDF-Spezifikation, in welcher die Datei erstellt wurde
- Body – Enthält Objektdaten, aus dem das Dokument besteht
- Cross-Reference Table – Enthält Positionen der Objekte in der Datei
- Trailer – Definiert Positionen des „Cross-Reference Table“ und von einzelnen speziellen Objekten im Datei-Body

Der Body in Abbildung 1 enthält unterschiedliche Objekttypen, zum Beispiel Seiten, Formularfelder oder Anlagen. Ein PDF-Dokument besteht stets aus einer festen Anzahl von Seiten, welche den Seiten im Ursprungsdokument entsprechen. Die Elemente einer Seite im Dokument stehen immer an derselben Position. Das Aussehen der Seiten ist durch eine Abfolge grafischer Elemente – einer Kombination von Texten, Grafiken und Bildern – bestimmt. In den Seiten des Dokuments befinden sich unter anderem Inhalt, Kommentare, Bilder und Schriften. Diese Elemente bilden eine komplizierte Struktur, welche das PDF-Dokument darstellen. Um die Speicherung des Dokuments zu optimieren, wird sowohl die Datenkomprimierung als auch das Erstellen komplexer Verbindungen zwischen den Daten angewendet. Hierdurch müssen Objekte, welche mehrfach im Dokument auftreten, nur einmal gespeichert werden [6, 8].

PDF-Dokumente können darüber hinaus Formularfelder enthalten. Ein Formularfeld ist ein Objekt, in das Felddaten abgelegt und gespeichert werden. Mögliche Formularfeldypen sind Textfeld, Kontrollkästchen, Auswahlknopf, Kombinationsfeld, Auswahlliste, Schaltfläche, Barcode- oder Unterschriftsfeld. Des Weiteren gibt es zahlreiche Funktionen wie zum Beispiel digitale Unterschriften oder das Aufrufen eines JavaScript-Codes [6, 8].

Außerdem ist es möglich, Eigenschaften in der PDF zu speichern. Dazu zählen die Identifizierung eines Dokuments sowie die Katalogisierung und Suche nach Dokumenten in Datenbanken. Beispiele für Daten sind Titel, Thema, Verfasser oder Schlüsselwörter [6, 8].

Vor- und Nachteile

Tabelle 1: Vor- und Nachteile des Dateiformates [6]

Vorteile	Nachteile
Plattformunabhäniges und freies Format	PDF-Datei nicht ohne Weiteres bearbeitbar
Integration rechtsgültiger elektronischer Signaturen (Unterschriften)	Großer Einfluss der verwendeten Auflösung auf Dateigröße
Komprimierte PDF-Daten, wenig Speicherplatzbedarf	Bearbeitung der Dokumente mittels besonderer Software
Zugriffs- und Bearbeitungsrechte können vergeben oder eingeschränkt werden	

3 Markup-Formate

3.1 Offline-Formate

Die hier vorgestellten Formate bzw. Technologien legen den Hauptfokus auf die Strukturierung von Dokumenten mit expliziten Deklarationen. Beispiele für solche Deklarationen sind Kapitel oder Abschnitte, in welchen ein Text unterteilt werden kann. Auf diese Weise wird zum Beispiel das Erstellen von Büchern, Artikeln und Dokumentationen vereinfacht.

3.1.1 Darwin Information Typing Architecture

Geschichte / Grund der Entwicklung

Darwin Information Typing Architecture (DITA) ist ein Dokumenten-Format, welches von dem Unternehmen IBM und der Organisation OASIS als Open-Source Architektur entwickelt wurde. Es steht als Dokumenttypdefinition (DTD) kostenlos zur Verfügung, welches auf XML basiert.

IBM benötigte in den 1990er Jahren ein neues Format, um für seine zahlreichen Produkte eine ausreichende Dokumentation erstellen zu können. Hierfür wurden eigene komplexe SGML-DTDs, wie IBMIDDoc entwickelt. Während der Weiterentwicklung der DTDs musste es möglich sein, ein hohes Maß an Wiederverwendung zu gewährleisten. Deshalb wurden andere Formate wie DocBook untersucht. Diese Formate erfüllten jedoch nicht die Anforderungen. So wurde entschieden, mit DITA ein neues Dokumenten-Format zu entwickeln. Mit DITA werden themenorientierte und informationstypisierte Inhalte in Form einer Quelle (single source) erstellt, welche wieder benutzt werden kann. Am 3. Mai 2005 wurde DITA 1.0 als OASIS-Standard verabschiedet und steht inzwischen seit 2010 in der Version 1.2 zur Verfügung [9]. DITA ist unter anderem im technischen Umfeld und bei Entwicklern sehr beliebt.

Technisches

DITA arbeitet nicht nach dem what-you-see-is-what-you-get (WYSIWYG)-Prinzip, sondern nach dem des what-you-see-is-what-you-asked-for (WYSIWYAF)-Prinzip. Darüber hinaus zeichnet sich das Format vor allem durch folgende Eigenschaften aus [9]:

- Themenzentrierung - Das Thema (Topic) ist der wichtigste Abschnitt. Weitere Strukturen, welche in der Hierarchie darüber stehen, sind in den meisten Fällen nur Zusatzinformationen. Innerhalb der Topics befinden sich die Sections mit Inhalt.
- Wiederverwendung - Das Ziel von DITA ist das Reduzieren von mehrfach auftretenden Informationen. Werden gleiche Informationen an mehreren Orten verwendet, verringert sich der Kopieraufwand.

In DITA gib es mehrere Elemente, welche näher erläutert werden [9]:

- Topic – Definition einer Informationseinheit mittels Titel und Inhalt. Diese darf nicht zu ausführlich sein, da ein Topic nur ein Thema oder eine Frage behandeln soll.
- Map – Sammlung und Organisation von Referenzen zu den Topics in Form eines Inhaltsverzeichnisses oder einer Gliederung.
- Spezialisierung – Erstellung neuer Informationstypen wie „structural types“ oder „new domains of information“. Voraussetzung ist die Wiederverwendung des größten Teils der Informationen. Damit soll der Aufwand für Austausch, Migration und Wartung minimiert werden.
- Integration – Erstellung neuer Dokumenttypen durch Kombination von Designmodulen.
- Anpassung – Ausgabe in unterschiedlicher Form ohne die Übertragung und den Austausch zu verschlechtern.

Aufbau

In Auflistung 1 [10] ist ein Minimal-Beispiel eines DITA-Dokuments dargestellt .

```
1 <topic id="maintaining" xml:lang="en-us">
2    <title>Maintaining</title>
3    <shortdesc> Kurzbeschreibung des Topics. </shortdesc>
4    <body>
5       <p> Text des Topics. </p>
6    </body>
7 </topic>
```

Auflistung 1: Dita-Datei `dita.xml`

Wie bereits erwähnt, basiert DITA auf XML. Daher kann man in einem DITA-Dokument direkt mit der Definition, in diesem Fall mit einem Topic, beginnen. Als erstes wird der Topic-Knoten definiert. Dieser enthält zusätzliche Metainformationen. Im Beispiel ist das eine ID, welche in der Sprache des Topic beschrieben wird. Im Topic-Knoten wird zunächst ein Titel definiert, welches in Zeile 2 zu sehen ist. Im Anschluss wird hier eine Kurzbeschreibung des Topic eingefügt. Darauf folgt der Body des Topics mit der eigentlichen Beschreibung, welcher ab Zeile 6 beginnt. In diesem Fall wurde der Text nochmals in einem Absatz eingeschlossen, welches in Zeile 7 zu erkennen ist. Neben diesen aufgeführten gibt es zahlreiche weitere Knoten und Auszeichnungen.

Vor- und Nachteile

Tabelle 2: Vor- und Nachteile des Dateiformates DITA [9, 11]

Vorteile	Nachteile
Verwendung des XML-Standards	Kein WYSIWYG
Erstellung und Auslieferung von Topics	Kein DTP-Ersatz
Kontextabhängige Informationen in verschiedenen Dokumenten gruppierbar	Nicht einfach, da viele Tags, strikte Struktur, viele kooperierende Tools
Inhalt und Format getrennt	Perfekte Druckergebnisse schwierig
Erweiterbar und auf individuelle Bedürfnisse und Anforderungen anpassbar	Keine vollständige Dokumentation
Läuft auf vielen Plattformen	Konvertierung nur in wenigen Formaten

3.1.2 DocBook

Geschichte / Grund der Entwicklung

DocBook ist ein Dokumenten-Format, welches im Jahr 1991 von der Gruppe Usenet entwickelt wurde. Dabei kam es in den Projekten HAL Computer Systems (IBM POWER architecture) und O'Reilly & Associates (Buchverlag, Erstellung von Büchern) zum ersten Mal zum Einsatz.

DocBook erleichtert das Erstellen von Büchern, Artikeln und Dokumentationen im technischen Umfeld. Sein offener Standard wird von der Organization for the Advancement of Structured Information Standards (OASIS) gepflegt. Außerdem ist DocBook in Open-Source-Projekten weit verbreitet. So verwendet zum Beispiel Linux Documentation Project die Dokumenttypdefinition (DTD) zum Erstellen der Howtos. Darüber hinaus verwenden KDE

und GNOME/GTK+ DocBook als grundlegendes Format für alle API-Dokumentationen sowie Benutzerhandbücher der Anwendungsprogramme. DocBook liegt in Version V5.1 vom November 2016 vor. Außerdem existieren für DocBook eine Reihe frei verfügbarer Tools, mit deren Hilfe HTML, PDF, Microsoft HTMLHelp und diverse andere Ausgabeformate erzeugt werden können. [12, 13].

Technisches

DocBook ist in einer für SGML und XML vorliegenden DTD entwickelt worden. Dabei wird bei Dokumenten nicht das Aussehen des Inhalts, sondern dessen Bedeutung beschrieben. Damit ist DocBook ein semantisches Format. Des Weiteren stellt DocBook eine große Anzahl von semantischen Element-Tags bereit. Sie sind in drei Kategorien unterteilt: Struktur, Block- und Inline-Elemente. Nicht alle dieser Elemente enthalten direkt Text [14].

Strukturelle Elemente geben Merkmale ihres Inhalts an. Das Buchelement gibt beispielsweise an, dass seine untergeordneten Elemente Teile eines Buches darstellen. Dazu gehören Titel, Kapitel, Glossare, Anhänge usw. [13].

Block-Elemente werden in der Regel mit einem Absatz vor und nach ihnen eingeleitet. Die meisten von ihnen können andere Blockelemente enthalten und viele können Text und Inline-Elemente enthalten. Beispiele für Blockelemente sind: Paragraphen, Listen, Seitenleisten, Tabellen und Blockzitate [13].

Inline-Elemente sind in der Regel ohne offensichtliche Unterbrechungen dargestellt. Die häufigste Unterscheidungsmerkmal der Inline-Elemente ist eine Schriftartänderung, aber sie können auch überhaupt keine visuelle Unterscheidung darstellen. Außerdem enthalten sie Text und eventuell andere Inline-Elemente aber beherbergen keine Block-Elemente. Sie werden verwendet, um Daten zu markieren. Einige Beispiele sind: Querverweise, Dateinamen, Befehle, Optionen, Indizes, Hervorhebung, Hyperlinks und Hochschriften sowie Glossarbegriffe [13].

Allerdings interpretieren DocBook-Prozessoren die Befehle unterschiedlich. Das heißt, bei einem Hervorhebungs-Tag, der eine Kursivschrift erzeugen soll, könnte von einem readerbasierter DocBook-Prozessor die Schriftgröße erhöhen oder ein textbasierter DocBook-Prozessor könnte fett statt kursiv verwenden [13, 15].

Das Layout von DocBook-Dokumenten wird über Stylesheets und DTDs gesteuert, welche in einer XSL gespeichert werden. Dabei stehen viele Vorlagen frei zur Verfügung, die von Institutionen oder Unternehmen erstellt worden sind. So kann ein Layout herausgesucht, Anpassungen vorgenommen und so auf den eigentlichen Inhalt konzentriert werden [13, 15].

Aufbau

In Auflistung 2 [16] ist ein Minimal-Beispiel eines DocBook-Dokuments dargestellt.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2   <book xml:id="simple_book" xmlns="http://...docbook" version="5.1">
3     <title>Very simple book</title>
4     <chapter xml:id="chapter_1">
5       <title>Chapter 1</title>
6       <para> Hello World! <emphasis>splendidly</emphasis>!</para>
7     </chapter>
8     <chapter xml:id="chapter_2">
9       <title>Chapter 2</title>
10      <para>Hello again, world!</para>
11    </chapter>
12  </book>
```

Auflistung 2: DocBook-Datei `docbook.xml`

Das hier abgebildete Dokument ist semantisch ein „Buch“. Dieses wurde in der Zeile 2 mit dem Tag `book` definiert. Das Buch hat einen „Titel“ und zwei „Kapitel“ mit jeweils eigenen „Titeln“. Diese „Kapitel“ enthalten „Absätze“ als Text, was in den Zeilen 3-7 zu erkennen ist.

Grundsätzlich ist das Buch das Wurzel-Element des Dokuments. Alle DocBook-Elemente befinden sich in einem XML-Namespace, sodass das Root-Element ein `xmlns-Attribut` hat, um den aktuellen Namespace festzulegen. Außerdem muss das Root-Element eine Version haben, welche die Version des DocBook-Formats angibt, auf dem dann das Dokument aufbaut.

Darüber hinaus muss ein Buch-Element einen Titel oder ein Info-Element mit einem Titel haben. Dieses Element muss vor jedem Kind-Strukturelement stehen. Nach dem Titel kommen die Kind-Strukturen, in diesem Fall zwei Kapitel-Elemente. Jedes von ihnen muss wiederum einen Titel besitzen. Sie enthalten `Para-Block`-Elemente, den freien Text und andere Inline-Elemente. Das ist in Zeile 6, also im zweiten Absatz des ersten Kapitels zu sehen.

Vor- und Nachteile

Tabelle 3: Vor- und Nachteile des Dateifromates DocBook [12, 17]

Vorteile	Nachteile
Verwendung des XML-Standards	Kein WYSIWYG
DocBook aktiv erweitert und gepflegt	Kein DTP-Ersatz
Stylesheets unterstützen Zielformate wie EPUB, HTML, Javahelp und XHTML	Nicht einfach, da viele Tags (HTML: 90, DocBook: 400), strickte Struktur, viele kooperierende Tools
Inhalt und Format getrennt	Perfekte Druckergebnisse schwierig
Läuft auf vielen Plattformen	Große komplexe Dokumente unübersichtlich

3.1.3 Electronic Publication

Geschichte / Grund der Entwicklung

Electronic Publication (EPUB) ist ein offener Standard für E-Books des International Digital Publishing Forum (IDPF). EPUB soll den älteren Standard Open eBook (OEB) bzw. Open eBook Publication Structure (OEBPS) ersetzen. OEB wurde von den zwei Unternehmen SoftBook Press (hauptsächlich) und NuvoMedia Inc. entwickelt, welche sich mit E-Book Systemen beschäftigen und für ihre Lesegeräte ein flexibles Format benötigten. OEB wurde in dem Jahr 1999 in der Version 1.0 veröffentlicht, die sogenannte Open eBook Publication Structure (OEBPS). 2002 wurde die Version 1.2 mit zahlreichen Formatempfehlungen aktualisiert. Im Jahr 2007 wurde OPS 2.0 unter dem heutigen Namen EPUB veröffentlicht. Seit Anfang 2017 ist das IDPF mit dem World Wide Web Consortizium (W3C) vereint. Die Weiterentwicklung von EPUB wird seitdem von einer Arbeitsgruppe des W3C fortgeführt [18, 19]. Der EPUB-Standard basiert auf einer Anzahl freier Standards. Dazu zählen beispielsweise XML, XHTML, DTBook, SVG, CSS, NCX, Dublin Core und ZIP. Ferner kommen weitere Bild-, Ton- und Video-Formate zum Einsatz, welche teilweise Lizenzen voraussetzen [18].

Technisches

Die Struktur von EPUB unterteilt sich in folgende Punkte [18, 20, 21]:

- Sichtbarer Inhalt – Open Publication Structure (OPS): Kümmert sich um den eigentlichen Inhalt. EPUB unterscheidet grundsätzlich zwischen den eigentlichen Inhaltsdokumenten und weiteren Dokumenten in anderen Formaten, welche in diese eingebettet werden können. Ab Version 3 werden Inhaltsdokumente entweder mit HTML5 oder SVG erstellt. Für die Stilvorlagen wird meistens CSS verwendet, um zum Beispiel die Farbe der Schrift festzulegen.
- Paketformat – Open Packaging Format (OPF): Strukturiert das Dokument. Die Struktur und die Aufnahme von Metadaten wird durch eine Stammdatei, welche die Dateiendung `.opf` besitzt, beschrieben. In dieser Datei werden im Element `metadata` die Metadaten nach dem Dublin-Core-Schema bereitgestellt. Außerdem listet das Element `manifest` alle weiteren im Archiv gespeicherten Dateien auf. Ein weiteres Element ist `spine` und stellt den Buchrücken des digitalen Buches bereit. Die Struktur dient hauptsächlich dazu, die Lesereihenfolge des Buches für das Darstellungsprogramm festzulegen. Dabei wird unterschieden, ob zwischen den Bestandteilen eines Buches ein linearer Lesefluss vorhanden ist. Dies kann bei jedem Inhaltsdokument separat angegeben werden. Mit weiteren optionalen Abschnitten können besondere Strukturen des Buches, wie Titelseite, Inhaltsverzeichnis, Index oder Glossar angegeben werden.
- Struktur des Archivs – OEBPS-Container-Format: Führt die aufgezählten Daten in eine einzelne Datei zusammen. Die Struktur des darin enthaltenen Dateisystems wird durch das OEBPS-Container-Format bereitgestellt. Die kompletten Teildaten und das Dateisystem werden zum Schluss in ein ZIP-Archiv mit der Dateiendung `.epub` zusammengeführt. Darin müssen mindestens folgende Daten enthalten sein [20]:
  - Erste Datei `mimetype` und Inhalt `application/epub+zip` im Wurzelverzeichnis
  - Verzeichnis `META-INF` mit der Datei `container.xml` mit Verweis auf die Stammdatei des Buches
  - Eigentliche Stammdatei
  - Inhaltsverzeichnis zur Navigation
  - Mindestens eine Inhaltsdatei
- Medienüberlagerungen: Stellen alternative Präsentationen wie Hörbücher bereit.
- Kanonische Fragment-Identifizierer: Realisiert Verweise auf beliebige Stellen im Buch.

Außerdem kann bei EPUB festgelegt werden, ob die Präsentation seitenbasiert oder rollbar sein soll und wie sich der Inhalt an den verfügbaren Platz anpassen soll. Hierfür gibt es zwei Möglichkeiten. Der erste Ansatz ähnelt SVG und wird durch das Festlegen eines Darstellungsbereiches und gegebenenfalls anschließender Skalierung (fixiert) durchgeführt. Der zweite Ansatz ähnelt XHTML, welcher durch automatische Aufteilung des Inhaltes auf den verfügbaren Platz ohne Skalierung (nicht fixiert) durchgeführt wird [18].

Mit der Verwendung von XML oder HTML ist es möglich, eine dynamische Anpassung des Textes an die jeweilige Bildschirmgröße des Lesers vorzunehmen. Sie eignen sich damit insbesondere für die Ausgabe auf mobilen Endgeräten. Für das Lesen von EPUB gibt es zahlreiche Programme, wie zum Beispiel Adobe Digital Editions (Windows, Mac), Aldiko (Android) und Calibre (Windows, Mac, Linux). Die eigenständige Erzeugung eines EPUB durch den Autor erfolgt unter anderem mit folgenden Programmen: Calibre (Windows, Mac, Linux), eBooksWriter (Windows) oder eCub (Windows, Mac, Linux). Sämtliche hier genannten Programme sind frei verfügbar [22].

Aufbau

In Auflistung 3 [21] ist ein Minimal-Beispiel eines EPUB-Archivs dargestellt. Zur Vereinfachung wurden die verschiedenen Dateien in eine Auflistung zusammengefasst.

```
# Container
  <container
    xmlns="urn:oasis:names:tc:opendocument:xmlns:container"
    version="1.0">
    <rootfiles>
      <rootfile
        full-path="inhalt.opf"
        media-type="application/oebps-package+xml"/>
    </rootfiles>
  </container>
# Stammdatei
  <package version="3.0"
    xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:opf="http://www.idpf.org
        /2007/opf" xmlns="http://www.idpf.org/2007/opf" unique-identifier="Id"
        >
    <metadata>
      <dc:identifier id="Id">6f2e78a1-c4f3-4895-b58b-347f92fb2d14</dc:
          identifier>
      <meta property="dcterms:modified">2013-10-26T17:27:34Z</meta>
      <dc:language>de</dc:language>
      <dc:title xml:lang="de">Hallo Welt</dc:title>
      <!-- bis hier notwendige Metainformationen, es folgen optionale: -->
      <dc:description xml:lang="de">Beispiel för ein Buch im Format EPUB 3.</
          dc:description>
  </package>
# Inhaltsverzeichnis
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ops="http://www.idpf.org/2007/ops"
      xml:lang="de">
  <head>
    <title>Inhaltsverzeichnis</title>
  </head>
  <body>
   <nav ops:type="toc">
    <h1>Inhaltsverzeichnis</h1>
    <ol>
      <li><a href="nav.xhtml">Inhaltsverzeichnis</a></li>
      <li><a href="inhalt.xhtml">Hallo Welt</a></li>
    </ol>
  </nav>
 </body>
</html>
# Eigentlicher Inhalt
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:ops="http://www.idpf.org
    /2007/ops" xml:lang="de">
  <head>
    <title>Hallo Welt</title>
  </head>
  <body>
    <h1>Hallo Welt</h1>
    <section ops:type="chapter">
      <h2>Titel des ersten Kapitels</h2>
      <p>Ein einfacher Absatz als Textbeispiel.</p>
      <p>...</p>
    </section>
    <section ops:type="chapter">
      <h2>Titel des zweiten Kapitels</h2>
      <p>...</p>
    </section>
  </body>
</html>
```

Auflistung 3: EPUB-Datei `epub.epub`

Zunächst wird in den Zeilen 1-10 die Container-Datei dargestellt, welche mithilfe von XML erstellt wird. In dieser wird in Zeile 7 auf die Stammdatei verwiesen, in welcher die Metadaten gespeichert werden.

Ab der Zeile 12 ist die Stammdatei abgebildet, welche auch von XML erstellt wird. Diese beinhaltet zum Beispiel das Datum einer Modifikation des Buches, welches in Zeile 16 zu erkennen ist. Außerdem wird hier angegeben, in welcher Sprache das Buch geschrieben worden ist und wie der Titel des Buches lautet. Zu erkennen ist dies in den Zeilen 17-19. Optional können weitere Metadaten angeben werden wie die Kurzbeschreibung des Buches oder der Publikation.

Im Anschluss wird ab Zeile 23 das Inhaltsverzeichnis dargestellt. Im Gegensatz zu den beiden vorigen Daten wird das Inhaltsverzeichnis in HTML erzeugt. Die Datei hat dabei einen Header, in welchem der Titel des Inhaltsverzeichnis angegeben wird und einen Body, in dem Verlinkungen zu den Inhaltsdateien hinterlegt werden. Dieses wird in den Zeilen 26-37 dargestellt.

In der letzten Datei befindet sich der eigentliche Inhalt, welcher in den Zeilen 40-56 zu erkennen ist. Der Inhalt wird erneut mit HTML erstellt und umfasst somit viele Gestaltungsmöglichkeiten des Textes. Dabei besitzt die Inhaltsdatei einen Header und einen Body. Im Header werden Metadaten wie der Titel angeben und im Body befindet sich der Inhalt. Dabei können Abschnitte mit einer Überschrift, zum Beispiel der ersten Ordnung, angegeben werden und darauf folgend eine Section, welche ein Kapitel einleitet. In der Section kann man wiederum weitere Unterabschnitte und Absätze definieren, welches in den Zeilen 46-50 demonstriert wird.

Vor- und Nachteile

Tabelle 4: Vor- und Nachteile des Dateifromates EPUB [23, 24]

Vorteile	Nachteile
Skalierbare Inhalte, d.h Anpassung an Bildschirmgröße	Fehlende Interoperabilität der Standards zwischen verschiedene Readern
Basiert auf freien Standards, geringe Kosten	DRM schränkt Handhabung ein
Transport zahlreicher Bücher mit einem Lesegerät	Fehlende Funktionen wie Wörterbuch, Vorlesefunktion und digitale Signatur

3.1.4 Latex

Geschichte / Grund der Entwicklung

Latex ist ein Softwarepaket, welches auf Tex basiert. Das Softwarepaket, welches verschiedene Makros enthält, soll dem Benutzer die Handhabung mit Tex erleichtern. Tex wurde von Donald E. Knuth während seiner Zeit als Informatik-Professor an der Stanford University entwickelt. Auf Tex basierend entwickelte Leslie Lamport Anfang der 1980er Jahre Latex. Er beendete 1990 mit Version 2.09 sein aktives Weiterentwickeln von Latex [25]. Seitdem entwickeln Autoren um Frank Mittelbach, Chris Rowley und Rainer Schöpf Latex weiter [26].

Technisches

Latex selbst besteht nur aus einem Basissatz von Makros. Darüber hinaus gibt es sehr viele Zusatzpakete, welche sich der Autor nachinstallieren kann, um die verschiedenen Anforde-

rungen abzudecken. Der Basisdatensatz bietet verschiedene Dokumentklassen wie `article`, `book`, `report`, `letter` oder `slides`. Diese wiederum stellen Strukturierungsmöglichkeiten wie Kapitel, Abschnitte und Möglichkeiten zur Querreferenzierung bereit. Außerdem können beispielsweise Literatur, Abkürzungsverzeichnisse oder Stichwortverzeichnisse automatisiert generiert werden [27]. Neben den Standardpaketen benötigt man hierzu ergänzende Software, welche die Erstellung erleichtern soll. Eine Beispiel-Anwendung ist BibLatex für das automatische Erstellen des Literaturverzeichnisses.

Des Weiteren arbeitet Latex nicht nach dem WYSIWYG-Prinzip, sondern das Dokument wird vor dem Ausführen in Textdateien definiert. Anschließend wird der Quellcode von Latex verarbeitet und in ein separates Dokumenten-Format, wie zum Beispiel PDF oder HTML, überführt. Das dabei von Latex generierte Layout gilt als sehr funktionell und sein Formelsatz als sehr ausgereift. Daher eignet sich Latex sehr gut für das Verfassen langer Arbeiten wie Abschlussarbeiten und Dissertationen, da solche Arbeiten oft an strenge Formvorgaben gebunden sind. Besonders in Mathematik und Naturwissenschaften eignen sich Zusatzpakete für die Erstellung von Formeln, da die Formelsetzung gegenüber üblichen Textverarbeitungssystemen leicht zu bewerkstelligen sind. Allerdings braucht es für die Bedienung von Latex einige Übungszeit, um mit dem System gegenüber üblichen Textverarbeitungssystemen sicher umgehen zu können. Dieser Aufwand rentiert sich vor allem für spätere Projekte mit ähnlichen Anforderungen. Der Vorteil besteht darin, dass die Grundstruktur bereits definiert ist und der Anwender schnell in den Schreibfluss übergehen kann [27].

Für die Benutzung von Latex werden Markups für die Dokumentenstruktur verwendet. Soll zum Beispiel in einem Dokument eine Überschrift erstellt werden, welche der ersten Ordnung entspricht, so wird diese mit dem Markup `\section{Einleitung}` definiert. Die Gestaltung des Abschnitt wird in den Klassen- oder Style-Dateien global festgelegt. Weiterhin werden für Latex optimierte Schriftarten in der Installation von Latex bereits mitgeliefert. Deshalb werden keine System-Schriftarten benötigt. Auch sind das Tex- und Latex-System unabhängig und können daher auf sämtlichen Plattformen wie Windows, OSX oder Linux verwendet werden [28]. Aufgrund einer großen und aktiven Community, seiner Stabilität, der freien Verfügbarkeit für viele Betriebssysteme und der Formelhandhabung sowie seiner Eigenschaften speziell für wissenschaftliche Arbeiten wird Latex vor allem an Universitäten und Hochschulen verwendet. Im Allgemeinen werden die Latex-Dokumente mithilfe einer Entwicklungsumgebung erstellt, wie zum Beispiel TexStudio oder TexMaker. In diesen Anwendungen wird die Benutzung von Latex nochmals erleichtert, da diese über verschiedene Menüs die Deklaration der Markups vereinfachen. Außerdem übernehmen sie die Übersetzung des Dokuments zum Beispiel in eine PDF.

Aufbau

Latex-Befehle beginnen immer mit einem Backslash (\) und können Parameter zwischen geschweiften ({}) sowie optionale Parameter zwischen eckigen Klammern ([]) enthalten (\befehl[optionaler parameter]parameter).

In Auflistung 4 [29] auf der nächsten Seite ist ein Minimal-Beispiel eines Latex-Dokuments dargestellt.

```
1  \documentclass{scrartcl}
2
3  \usepackage[utf8]{inputenc}
4  \usepackage[T1]{fontenc}
5  \usepackage{lmodern}
6  \usepackage[ngerman]{babel}
7  \usepackage{amsmath}
8
9  \title{Ein Testdokument}
10 \author{Otto Normalverbraucher}
11 \date{5. Januar 2004}
12 \begin{document}
13
14 \maketitle
15 \tableofcontents
16 \section{Einleitung}
17
18 Hier kommt die Einleitung. Ihre Ueberschrift kommt
19 automatisch in das Inhaltsverzeichnis.
20
21 \subsection{Formeln}
22
23 \LaTeX{} ist auch ohne Formeln sehr nötzlich und
24 einfach zu verwenden. Grafiken, Tabellen,
25 Querverweise aller Art, Literatur- und
26 Stichwortverzeichnis sind kein Problem.
27
28 Formeln sind etwas schwieriger, dennoch hier ein
29 einfaches Beispiel.
30 \begin{align}
31 E &= mc^2                    \\
32 m &= \frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}}
33 \end{align}
34 \end{document}
```

Auflistung 4: Latex-Datei `latex.tex`

In Zeile 1 wird definiert, welche Art von Text im Dokument benutzt werden soll. Hier wird eine KOMA-Script-Klasse `scrartcl` angegeben, um Latex mitzuteilen, dass es sich um einen deutschen Text handelt. Außerdem ist diese Klasse für kürzere Artikel gedacht. Andere Klassen wie `scrreprt` werden für längere Berichte und `scrbook` für Bücher verwendet. Diese Klassen definieren unter anderem in welcher Hauptgliederungsebene der Text beginnt. So beginnen `scrbook` und `scrreprt` mit Kapitel (\chapter), `scrartcl` mit Abschnitt `(\section)`.

Mit dem Befehl `(\usepackage)` in den Zeilen 3-7 werden verschiedene Latex-Pakete in das Dokument eingebunden, zum Beispiel `(\inputenc)` mit der Option UTF-8. Dieses weist Latex an, wie der zu erstellende Text kodiert werden soll.

Mit dem Befehl `(\begin {Document})` in Zeile 12 beginnt der eigentliche Inhalt des Dokuments. Der Text, der nun folgt, wird später in der PDF angezeigt.

In Zeile 14 wird ein Titel ausgegeben, in Zeile 16 wird eine Überschrift erster Ordnung definiert. Danach erfolgt ab Zeile 23-30 der Text, welcher ohne jegliche Befehle auskommt. In Zeile 31-34 wird eine mathematische Formel definiert.

Vor- und Nachteile

Tabelle 5: Vor- und Nachteile des Dateiformates Latex ([30])

Vorteile	Nachteile
Kostenlose Verfügbarkeit	Schwer zu erlernen, da kein WYSIWYG-Prinzip
Plattformunabhängig und -übergreifend, d.h. das Quelldokument kann nahezu von jedem Computersystem betrachtet, editiert und gedruckt werden	Kein offizieller Support, jedoch zahlreiche Publikationen und sonstige Anwendervereinigungen, die bei Fragen helfen
Gute typographische Ergebnisse, d.h. die Dokumente erfüllen höchste Anforderungen	Komplizierte Modifikation der Standardlayouts, d.h. intensives Einarbeiten für eigene Dokumentvorlagen
Guter mathematischer Formelsatz, relativ unkomplizierte Erstellung der Formeln	Syntax-Fehlermeldungen können irritieren, da ungenaue Positionsangabe der Fehler
Logische statt physische Formatierung, d.h. Latex übernimmt komplette Formatierung, während der Anwender überwiegend nur die Struktur des Dokuments bestimmt	Kompatibilität mit dem „Standard“-Wordformat
Stabilität, d.h. eines der stabilsten Programmpakete, Datenverlust sehr selten	Installation von Latex nicht immer ganz einfach

3.2 Dokumenten-Formate

Die Dokumenten-Formate bzw. -Technologien legen den Hauptfokus auf das leichte Erstellen von Dokumenten wie Artikel, Publikationen oder andere Texte ohne expliziten Deklarationen, welche en Text in genaue Abschnitte unterteilen.

Hierbei gibt es zum einen Formate, welche nach dem Prinzip des WYSIWYG arbeiten. Andere Formate müssen erst kompiliert und im Anschluss in ein anderes Layout überführt werden.

3.2.1 Open Document Format for Office Applications

Geschichte / Grund der Entwicklung

Open Document Format for Office Applications (ODF) kann mit Office-Anwendungen erstellt werden, zum Beispiel mit OpenOffice oder LibreOffice. Das ODF-Dateiformat basiert auf dem Standard-Dateiformat von OpenOffice.org Version 1 und wurde ursprünglich von Sun Microsystems entwickelt. Später wurde es von Organization for the Advancement of Structured Information Standards (OASIS) spezifiziert und weiterentwickelt. OASIS ist eine internationale nicht-gewinnorientierte Organisation, die sich mit der Weiterentwicklung von e-Business- und Web-Service-Standards beschäftigt. 2006 wurde ODF als internationale Norm ISO/IEC 26300 veröffentlicht [31].

ODF wurde unter anderem entwickelt, um einen offenen Standard zur Verfügung zu stellen. Damit können Dokumente verschiedener Programme verarbeitet und untereinander Daten ausgetauscht werden. Ein möglichst breiter Einsatz von ODF wird erreicht, indem die Hindernisse zur Implementierung möglichst gering gehalten werden. Als Folge steht ODF in zahlreichen

Programmen zur Verfügung. Eine Besonderheit von ODF ist die konsequente Verwendung bereits vorhandener Standards. Somit wird eine Wiederverwendung von Technologien erreicht. Zum Beispiel basiert ODF auf XML und verwendet unter anderem den W3C-Standard SVG, welches zur Beschreibung von Vektorgrafiken verwendet wird, sowie den Standard Dublin Core für Metadaten [32]. Die Popularität von ODF wächst ständig und wird inzwischen von vielen Privatpersonen sowie in zahlreichen Unternehmen und Institutionen verwendet. So beschloss zum Beispiel der IT-Rat der Bundesregierung 2008, dass ODF in der Bundesverwaltung schrittweise eingesetzt werden soll [33].

Technisches

Bei einer ODF-Datei handelt es sich um eine Container-Datei, welche wie ein ZIP-Archiv funktioniert. Das Archiv beinhaltet einen Ordner `meta-inf` mit einer Datei `manifest.xml`, worin alle weiteren Dateien im ODF-Container mit ihren jeweiligen MIME-Typen angegeben sind. Diese sind normalerweise in XML erstellt und enthalten die eigentlichen Daten wie Dokument-Struktur, Dokument-Inhalt, Dokument-Style, Dokument-Einstellungen usw. Des Weiteren können eingebettete Multimedia-Dateien wie Bild-, Film- oder Musik-Dateien sowie ein Vorschaubild (Thumbnail) der Datei enthalten sein. Für mathematische Formeln wird ein Teil der Auszeichnungssprache MathML verwendet, welche ebenfalls auf XML basiert. Darüber hinaus ist es möglich, ODF beliebig mit anderen XML-basierten Sprachen zu erweitern oder zahlreiche Plugins in den Office-Anwendungen nachzuinstallieren. ODF wird überwiegend in Programmen genutzt, die nach dem WYSIWYG-Prinzip funktionieren [31].

Die nachfolgende Tabelle stellt eine Übersicht der wichtigsten Dateiendungen der ODF-Dateien zusammen:

Tabelle 6: Übersicht Dateiendungen (Von: [31])

Dateiendung	**Name**	**Modul in OpenOffice / LibreOffice**
.odt	OpenDocument Text	Text-Verarbeitung
.ods	OpenDocument Spreadsheet	Tabellen-Kalkulation
.odp	OpenDocument Presentation	Präsentation
.odg	OpenDocument Graphics	Zeichnung
.odf	OpenDocument Formula	Formel-Editor

Aufbau

In Auflistung 5 [Autor] ist als Beispiel die Datei `manifest.xml` einer ODT-Datei abgebildet.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <manifest xmlns="urn:oasis:names:tc:opendocument:xmlns:manifest:1.0">
3    <file-entry media-type="application/vnd.oasis.opendocument.text" full-path="
       /"/>
4    <file-entry media-type="application/vnd.sun.xml.ui.configuration" full-path=
       "Configurations2/"/>
5    <file-entry media-type="image/png" full-path="Pictures/10000000000001
       E800000118B5A37F3F.png"/>
6    <file-entry media-type="" full-path="Pictures/"/>
7    <file-entry media-type="text/xml" full-path="content.xml"/>
8    <file-entry media-type="text/xml" full-path="styles.xml"/>
9    <file-entry media-type="text/xml" full-path="meta.xml"/>
10   <file-entry media-type="" full-path="Thumbnails/thumbnail.png"/>
11   <file-entry media-type="" full-path="Thumbnails/"/>
12   <file-entry media-type="text/xml" full-path="settings.xml"/>
13 </manifest>
```

Auflistung 5: Manifest-Datei `manifest.xml`

Die Manifest-Datei beinhaltet alle notwendigen Daten des ODF-Archivs. So wird in Zeile 2 angegeben, dass es sich um ein ODF-Manifest und in Zeile 3 um ein ODT-Datei-Archiv handelt. Ab Zeile 4 werden verschiedene Dateien aufgelistet, welche zum Beispiel den Style des Dokuments festlegen. In Auflistung 6 [Autor] ist ein Ausschnitt der `content.xml`-Datei abgebildet, in welcher der Dokumenteninhalt gespeichert wird .

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <office:document-content xmlns:office="urn:oasis:names:tc:opendocument:xmlns:
       office:1.0" xmlns:text="urn:oasis:names:tc:opendocument:xmlns:text:1.0"><
       office:scripts/><office:font-face-decls>
3  <style:font-face style:name="Tahoma1" svg:font-family="Tahoma"/><style:font-
       face style:name="Times New Roman" svg:font-family="&apos;Times New Roman&
       apos;"><style:style style:name="P2" style:family="paragraph" style:parent-
       style-name="Standard"></style:style></office:scripts>
4      <office:body>
5          <office:text>
6              <text:sequence-decls>
7                  <text:sequence-decl text:display-outline-level="0" text:name="
                       Illustration"/>
8                      <text:sequence-<text:p text:style-name="P1">Generelles
                           Standardisierungsproblem einer Auszeichnungssprache
9                          </text:p><text:p text:style-name="P1"/><text:p text:
                               style-name="P2">
10                         OpenDocument als Dokumentenformat, das ...
11                     </text:p>
12         </office:text>
13     </office:body>
14 </office:document-content>
```

Auflistung 6: Ausschnitt aus der Datei `content.xml`

Zunächst werden Metainformationen wie die Schriftart des Textes definiert, welches in Zeile 3 beispielhaft mit dem Befehl `style:name` gezeigt wird. Ab Zeile 4 beginnt der `office:Body`, in welchem der Text eingefasst wird. Mit dem Befehl `text:p` in Zeile 9 beginnt ein Paragraphen-Abschnitt.

Vor- und Nachteile

Tabelle 7: Vor- und Nachteile des Dateiformates ODF ([31, 34, 35])

Vorteile	Nachteile
International standardisiertes Format, d.h. weltweit einheitlich und kompatibel	Metainformationen werden in verschiedenen Programmen unterschiedlich interpretiert
Offenes und herstellerunabhängiges Format, d.h. die Spezifikationen stehen öffentlich zur Verfügung	Beschränkung auf Standards, die alle anzeigenden Programme nutzen, zum Beispiel unterstützt OpenOffice nicht alle MS-Excel Formeln und umgekehrt
Freies Format, d.h. keine Lizenzgebühren für die Benutzung des Formates	Unflexibles Layout (Papierformat, Absatz- und Schriftmerkmale etc.)
Zukunftssicheres Format, d.h. ODF nicht an bestimmte Software-Versionen gebunden	Darstellungsprobleme zwischen verschiedenen Betriebssystemen oder Druckertreibern
Durch einen ASCII-Editor auslesbar	
Programmunabhängiges Format, d.h. nicht an bestimmtes Programm gebunden	

3.2.2 Office Open Extensible Markup Language

Geschichte / Grund der Entwicklung

Office Open XML (OOXML) wurde von Microsoft im Jahr 2006 entwickelt und zuerst in Office 2007 zur Verfügung gestellt. Das Format OOXML dient zur Speicherung von Bürodokumenten auf XML-Basis, welches den Daten- und Dateienaustausch zwischen verschiedenen Büroanwendungen ermöglicht [36]. Da viele Unternehmen und Behörden zunehmend auf frei verfügbare Standards setzen, um die Verfügbarkeit und Lesbarkeit in der Zukunft zu sichern, hat Microsoft eine Normierung für OOXML beantragt. So ist OOXML frei verfügbar und kopierbar, was am 7. Dezember 2006 von der Ecma-International-Hauptversammlung als Ecma-Standard 376 genehmigt wurde [37].

Grund der OOXML-Entwicklung war, dass Microsoft das bereits existierende ODF-Format (siehe Abschnitt 3.2.1) ablehnte, weil es nicht geeignet war, alle bereits verfügbaren Microsoft-Office-Dokumente wiederzugeben. ODF basierte unter anderem zu sehr auf dem Dateiformat und damit auf der Vorgehensweise und dem Funktionsumfang von OpenOffice [38].

Technisches

Inzwischen gibt es verschiedene OOXML-Versionen, welche in den Office-Produkten unterschiedlich implementiert wurde [39]. In den Office-Paketen ab 2007 wurden Änderungen vorgenommen, die der nachfolgenden Liste zu entnehmen sind:

- Office 2007 liest und schreibt in „ECMA“
- Office 2010 und höhere Versionen lesen und schreiben in „Transitional“
- Office 2013 liest und schreibt nicht in „Strict“, jedenfalls nicht gemäß ISO/IEC 29500

So meint Michael Meeks, einer der Kernentwickler der Open Source Office-Suite LibreOffice, „. . . dass niemand ISO OOXML Strict in der freien Wildbahn verwendet. Wenn eine Firma

oder Verwaltung verschiedene Versionen von MS Office einsetzt, incl. Office 2007 oder 2010, dann erfüllen Dokumente, die von mehreren Leuten ausgetauscht wurden, nicht die strikte Variante von OOXML, wie in ISO/IEC 29500 vorgeschrieben“ [39]. Des Weiteren kann der Benutzer an der Dateiendung nicht erkennen, um welche OOXML-Version es sich handelt. Außerdem wird OOXML überwiegend in Programmen genutzt, die nach dem WYSIWYG-Prinzip funktionieren.

Aufbau

OOXML besteht aus dem Container-Format Open Packaging Conventions und einer Reihe von XML-basierten Auszeichnungssprachen, welche die OOXML-Dateien widerspiegeln. Dabei werden die Dokumente in Packages gespeichert, die dem Container-Format entsprechen. Das Package arbeitet nach dem Prinzip eines ZIP-Archivs, wie es auch ODF (siehe 3.2.1) handhabt, und speichert alle Bestandteile (Parts und Items) in sich ab. Im Archiv liegen die einzelnen Bestandteile (Parts bzw. Bausteine) wie zum Beispiel Bilder, Grafiken, Texte etc., während Items Metadaten beschreiben. Sie legen fest, wie die einzelnen Bestandteile des Dokuments zusammengestellt und dargestellt werden. Dabei wird bei den Items in Relationship Items und Content-Type Items unterschieden [36].

Die Relationship Items legen dar, welche Beziehungen die einzelnen Teile untereinander haben, d. h. wie die einzelnen Komponenten des Dokuments zusammengefügt werden müssen. Content-Type Items hingegen beschreiben den Content-Type der einzelnen Komponenten, d. h. wie die einzelnen Komponenten dargestellt werden müssen. Jede OOXML-Datei besteht immer aus einem `mainpart` und gegebenenfalls aus weiteren Komponenten, die mit `mainpart` über ein Relationship Item verbunden werden. Allerdings hängt der Aufbau, Name und Pfad innerhalb der ZIP-Datei vom OOXML-Dokumententypen ab. Ein Textverarbeitungsdokument ist anders strukturiert als ein Tabellenkalkulationsdokument [36, 40, 41].

Ein OOXML-Textverarbeitungsdokument hat in seiner minimalen Version in seiner Wurzel immer eine XML-Datei namens `/[Content_Types].xml` sowie drei Verzeichnisse `/_rels`, `/docProps` und ein Verzeichnis mit den eigentlichen Dokumentdaten [36, 40, 41].

- Die `[Content_Types].xml`-Datei enthält eine Auflistung der Dateien der Zip-Datei.
- Das `_rels`-Verzeichnis speichert die Abhängigkeiten zwischen den einzelnen Parts pro Part in einer eigenen Datei. Werden zum Beispiel in einem Textverarbeitungsdokument Text und Bilder eingefügt, sind die Verweise zu den Bildern in den Dateien `/word/document.xml` und `/word/_rels/document.xml` gespeichert.
- Das `docProps`-Verzeichnis enthält die Daten `core.xml, app.xml und custom.xml`, welche diverse Metadaten wie Autor, Speicherdatum etc. im Metadatenstandard Dublin-Core gespeichert sind.
- Das Dokumentdatenverzeichnis enthält die eigentlichen Dokumentdaten. Zum Beispiel ist in einem Textverarbeitungsdokument die Datei `document.xml` enthalten.

In Auflistung 7 [Autor] ist ein Ausschnitt aus der Datei `document.xml` abgebildet.

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <w:document
3     xmlns:o="urn:schemas-microsoft-com:office:office"
4     <w:body><w:p><w:pPr><w:pStyle w:val="Normal"/><w:rPr><w:b/><w:b/><w:bCs/><
          w:sz w:val="30"/><w:szCs w:val="30"/></w:rPr></w:pPr><w:r><w:rPr><w:b/
          ><w:bCs/<w:t xml:space="preserve">OpenDocument als Dokumentenformat,
          das auf einer Auszeichnungssprache basiert, enthölt Metainformationen,
           die vom darstellenden Programm ausgewertet bzw. interpretiert werden
          mössen (engl. ?Rendering?, siehe analog dazu HTML-Rendering). </w:t></
          w:r></w:p><w:sectPr><w:type w:val="nextPage"/><w:pgSz w:w="11906" w:h=
          "16838"/><w:pgMar w:left="1134" w:right="1134" w:header="0" w:top="
          1134" w:footer="0" w:bottom="1134" w:gutter="0"/><w:pgNumType w:fmt="
          decimal"/><w:formProt w:val="false"/><w:textDirection w:val="lrTb"/></
          w:sectPr>
5         </w:body>
6 </w:document>
```

Auflistung 7: Dokument-Datei `document.xml`

Zunächst werden in Zeilen 1-3 einige Metadaten definiert, welche zum Beispiel aussagen, um welches Dokument es sich handelt. Danach wird in Zeile 4 der Dokumentenbody definiert, in welchem sich der Text und deren Formatierungen sowie Header- und Footer-Definitionen befinden. Zum Abschluss werden in Zeile 5 der Body und in Zeile 6 das Dokument geschlossen.

Vor- und Nachteile

Tabelle 8: Vor- und Nachteile des Dateiformates OOXML ([42, 43, 39, 44])

Vorteile	Nachteile
Meist verbreitetstes Dokumenten-Format	Spezifikation umfasst mehr als 6000 Seiten, deren vollständige Umsetzung laut Kritikern nur durch Microsoft möglich ist
Sehr gute Implementierung in den MS Office Paketen	Verschiedene implementierte Versionen in den MS-Office-Paketen
Großer Funktionsumfang	Kein Verwendung der Standards MathML oder SVG, sondern Eigenentwicklungen
Einfache Integration von Geschäftsinformationen in Dokumente	Norm widerspricht ISO-Norm für Darstellung von Datum und Uhrzeiten oder Sprachabkürzungen
Offenes Format und gebührenfrei	Microsoft übt Druck und Kritik auf Politiker aus, um die Benutzung der konkurrierenden OpenDocument-Formate zu verhindern
Stabilität – vereinfachter Datenaustausch zwischen Microsoft Office-Anwendungen und Unternehmenssystemen	Erschwerte Entwicklung von Konvertierungswerkzeugen für OOXML, da entsprechende XSLT-Transformatoren nicht vollständig zur Verfügung stehen
Effizienz – ZIP und Komprimierungstechnologien zum Speichern von Dokumenten	Fehlerhafte Kompatibilität, weil älteren Formate weiter unterstützt werden

3.2.3 Rich Text Format

Geschichte / Grund der Entwicklung

Rich Text Format (RTF) ist ein Dateiformat zur Erstellung von Texten, welches 1987 von Microsoft entwickelt wurde. Es liegt seit 2007 in Version 1.9.1 vor.

RTF wird für den Austausch zwischen unterschiedlichen Textverarbeitungsprogrammen verschiedener Hersteller und Betriebssystemen verwendet. Auch kann das Format in zahlreichen Programmen verwendet werden, die RTF unterstützen. Ebenso kann es mit ANSI, PC-8, Macintosh, und IBM PC character sets umgehen [45, 46].

Technisches

RTF-Dateien sind technisch gesehen reine Textdateien, welche über Anweisungen ihr Erscheinungsbild verändern und somit daher mit einer Markup-Sprache arbeiten. Allerdings ist eine Layouttreue nicht gewährleistet, obwohl viele Formatierungsmerkmale erhalten bleiben. Je einfacher der Text formatiert ist, desto besser wird er in anderen Programmen dargestellt. Problematisch sind Bilder, Textrahmen und andere Objekte, da das Layout beim Öffnen in anderen Programmen fehlerhaft sein kann. Des Weiteren werden Schriftarten normalerweise nicht in RTF integriert und sollten deswegen auch auf dem Zielsystem installiert sein, um mögliche Fehler auszuschließen.

Ab Version 1.9.1 ist es möglich, XML Markup-Tags, SmartTags und Mathematik-Elemente in RTF zu integrieren. Damit werden die Dokumentgestaltungsmöglichkeiten erweitert. Des Weiteren werden Microsoft Object Linking and Embedding (OLE)-Objekte oder Macintosh Edition Manager subscriber-Objekte verwendet, um zum Beispiel Tabellen oder Charts aus der Tabellenkalkulation in RTF-Dokumenten zu integrieren. Allerdings sollten andere Programme OLE interpretieren können, da sonst die Inhalte nur als Bitmap oder gar nicht angezeigt werden [47]. Andere Formate, wie zum Beispiel ODF (siehe 3.2.1) als Dateiaustauschformat, sind besser geeignet als das RTF-Dateiaustauschformat, da die Gestaltungsmöglichkeiten vielseitiger sind [45]. Außerdem wird RTF überwiegend in Programmen genutzt, die nach dem WYSIWYG-Prinzip funktionieren.

Aufbau

In Auflistung 8 [48] ist ein Minimal-Beispiel eines RTF-Dokuments dargestellt.

```
1 {\rtf1
2 Hallo Welt!
3 \line
4 {\i Dies} ist \b{\i ein
5 \i0 formatierter \b0Text}.
6 \par
7 \i0 Das \b0Ende.
8 }
```

Auflistung 8: RTF-Datei `datei.rtf`

Zunächst definiert `{\rtf1` in Zeile 1 das RTF-Dokument. Im Anschluss beginnt der Text mit seinen Formatierungen. Eine Leerzeile kann mit `\line` erzeugt werden, was beispielsweise in Zeile 3 vorgenommen wird. Mit dem Befehl `{\i <Text>}` in Zeile 4 wird der darin enthaltene Text kursiv und mit `\b {\i <Text>}` in Zeile 4 fett und kursiv angezeigt. Schriftarten können zu Beginn einer RTF-Datei angegeben und dann verwendet werden [45]. Die Darstellung der oben gezeigten Formatierung sieht wie folgt aus:

Hallo Welt!
Dies ist ***ein*** **formatierter** Text.
Das Ende.

Vor- und Nachteile

Tabelle 9: Vor- und Nachteile des Dateiformates RTF ([45, 49])

Vorteile	Nachteile
Einfache Syntax	Keine Layouttreue, Verlust von Formatierungen
In vielen Programmen und Plattformen lesbar	Keine Mitnahme von Schriften
Einfach zu verwenden; von MS Word, AppleWorks, Corel OppenOffice u.a. unterstützt	RTF Standard von anderen Formaten verdrängt
Unterstützt keine Makros, keine Virenübertragung	Fehlende Unterstützung von Makros

3.3 Online-Formate

Die Formate bzw. Technologien der Online-Formate legen den Hauptfokus auf den Online-Einsatz in Foren, Blogs, Webseiten oder ähnlichen Gebieten. Dort vereinfachen sie das Schreiben oder die Gestaltung von Texten. Einige Formate strukturieren den Text, andere übernehmen dessen Gestaltung und andere wiederum sind für die Formatierung zuständig.

3.3.1 Bulletin Board Code

Geschichte / Grund der Entwicklung

Bulletin Board Code (BBCode) ist eine vereinfachte Auszeichnungssprache, welche an HTML angelehnt ist. Mit BBCode können Nachrichten in HTML-ähnlicher Weise formuliert werden. Sie wird hauptsächlich zur Erstellung von Forenbeiträgen oder Blogs verwendet. BBCode wurde erstmals 1998 in der Webforensoftware Ultimate Bulletin Board (UBB) Version 3 eingeführt und wird manchmal auch als UBBCode bezeichnet. Mit BBCode wird dem User die Möglichkeit gegeben, von ihm selbst erstellten Text zu formatieren [50].

Technisches

Durch den einfacheren Syntax-Aufbau ist es für den Nutzer nicht möglich, das Layout der Seite zu verändern. Damit können auch User, die keine HTML-Kenntnisse besitzen, BBCode benutzen. Normalerweise wird BBCode in der Form `[tag]...[/tag]` in den Text eingefügt. Ein Skript wandelt die benutzten `[tag]`'s in HTML-Code um [50].

Aufbau

In Auflistung 9 [51] ist ein Minimal-Beispiel eines BBCode-Dokuments dargestellt.

```
1  Auszeichnungselement
2      [elementname] [/elementname]
3
4  einfache Schriftformatierung
5      [b]fett[/b]
6      [i]kursiv[/i]
7      [u]unterstrichen[/u]
8      [s]durchgestrichen[/s]
9      [center]zentriert[/center]
10
11 Farbigen Text
12     [color={color}]{text}[/color]
13     [style color={color or hex}]{text}[/style]
14
15 Gliederung
16     ... lorem ipsum
17     Leerzeile
18     dolor ...
19
20 Bilder
21     [img]example.com/bild.jpg[/img]
22
23 Liste
24     [list]
25     [*]Punkt
26     [*]Punkt
27     [/list]
```

Auflistung 9: BBCode-Datei `bbcode.bb`

BBCode benötigt keine spezielle Struktur. Der Autor beginnt mit dem Erstellen eines Forenbeitrags und kann während des Schreibens die Formatierungen vornehmen.

Viele Befehle, die in BBCode zur Verfügung stehen, funktionieren wie HTML. Beispielsweise steht `[b]fett[/b]` für das Anzeigen eines fett geschriebenen Textes, `[i]kursiv[/i]` für einen kursiven Text und `[u]unterstrichen[/u]` für einen unterstrichenen Text. Dies wird in den Zeilen 5-9 demonstriert. Texte können des Weiteren auch farbig gestaltet werden. Hierzu wird zunächst eine Farbe festgelegt und anschließend der Style auf den Text angewendet. Diese Prozedur wird in den Zeilen 12-13 dargestellt. Eine Gliederung wird mit drei aufeinander folgenden Punkten eingeleitet und mit drei Punkten beendet, welches in Zeile 15-18 zu sehen ist. Will man dagegen ein Bild anzeigen, wird einfach eine URL, die auf das Bild zeigt, zwischen dem Tag `img` eingefügt. Für eine nummerierte Liste gibt es weitere spezielle Auszeichnungen, siehe Zeilen 23-27. Bei den hier demonstrierten Auszeichnungselementen handelt es sich lediglich um einen kleinen Teil der Elemente, die dem User zur Verfügung stehen.

Vor- und Nachteile

Tabelle 10: Vor- und Nachteile des Dateiformates BBCode

Vorteile	Nachteile
Einfache Syntax	Hauptsächlich im Foren-Umfeld
Sicherheitsprobleme, die der Einsatz von HTML mit sich bringt, werden umgangen	Spezielle Parser für Umwandlung benötigt
Eigene Formatierungen durch User	Im Sprachumfang gegenüber HTML deutlich unterlegen

3.3.2 Hypertext Markup Language

Geschichte / Grund der Entwicklung

Hypertext Markup Language (HTML) ist eine textbasierte Auszeichnungssprache zur Strukturierung digitaler Dokumente wie Texte mit Hyperlinks, Bildern und anderen Inhalten.

Die Idee von HTML entstand im Jahr 1989 als Forscher Forschungsergebnisse mit anderen Mitarbeitern der Europäischen Organisation für Kernforschung (CERN) teilen und von beiden Standorten in Frankreich und der Schweiz aus zugänglich machen wollten. Im CERN wurde daraufhin ein Projekt ins Leben gerufen, welches eine Lösung für dieses Problem entwickeln sollte. 1992 wurde als Ergebnis die erste Version mit dem Namen HTML veröffentlicht [52].

Technisches

HTML ist ein wichtiger Bestandteil des heutigen World Wide Web (WWW). Webseiten und Applikationen liefern mit HTML eine semantische Beschreibung des Inhalts wie Texte, Bilder, Hyperlinks und geben dem Dokument eine Struktur. Des Weiteren ist es möglich, Metadaten wie die Sprache des Textes oder den Namen des Autors in die HTML-Dokumente zu integrieren. Für die Präsentation, also die Darstellung des Inhaltes, ist HTML nicht zuständig, sondern Cascading Style Sheets (CSS), welches in dieser Ausarbeitung nicht weiter behandelt wird [4]. HTML wird vom World Wide Web Consortium (W3C) und der Web Hypertext Application Technology Working Group (WHATWG) ständig weiterentwickelt und steht zurzeit in der Version 5.1 zur Verfügung. Sämtliche Webbrowser und andere Layout-Engines, welche auf dem Markt bereit gestellt werden, unterstützen diese neuste Version [53].

HTML5 löst die früheren Versionen wie HTML4 und Extensible Hypertext Markup Language (XHTML) ab, die noch in geringem Umfang verwendet werden [5]. In HTML5 werden die jeweiligen Vorteile der bisherigen HTML-Versionen vereint, um eine einfachere Handhabung für die Dokumenterstellung zu erreichen. Außerdem ist es möglich, HTML5 mit vielen weiteren Komponenten wie zum Beispiel JavaScript zu erweitern. Des Weiteren arbeitet HTML nicht nach dem WYSIWYG-Prinzip, sondern das Dokument wird vor dem Ausführen erstellt und anschließend in einem separaten Programm ausgeführt [4, 53].

Aufbau

In Auflistung 10 [Autor] ist ein Minimal-Beispiel eines HTML-Dokuments dargestellt.

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title></title>
5      <meta content="">
6      <style></style>
7    </head>
8    <body>
9      <h1>Eine Ueberschrift</h1>
10     <p>Ein Textabsatz, der ein <em>betontes</em> Wort enthölt.</p>
11     Eine Textzeile,<br>die hier fortgesetzt wird.
12     <img src="E-Mail-Button.jpg" alt="E-Mail">
13   </body>
14 </html>
```

Auflistung 10: HTML-Datei `datei.html`

Ein HTML-Dokument besteht aus drei Teilen [4]:

- Der Dokumenttypdeklaration **(Doctype)**, welche sich am Anfang der Datei befindet und die verwendete Dokumenttypdefinition (DTD) angibt, wie zum Beispiel HTML.
- Dem HTML-Kopf **(HEAD)**, welcher primär dokumentarische Informationen enthält, die für den Anwender nicht sichtbar sind.
- Dem HTML-Körper **(BODY)**, in welchem sich der eigentliche Inhalt befindet, der angezeigt werden soll.

Zunächst wird in Zeile 1 dem Dokument mitgeteilt, dass es sich um ein HTML-Dokument handelt. Danach ist in Zeile 3-7 der Header definiert, in welchem zum Beispiel Metadaten wie Titel oder Content-Beschreibung enthalten sein können. Außerdem befinden sich im Header oft Style-Informationen, die im Body angewendet werden sollen. Ab Zeile 8 beginnt der eigentliche Inhalt des Dokuments, welcher mit dem Body-Tag eingeleitet wird. Mit SGML-Elementen wird der Text strukturiert. Ein Tag beginnt immer mit einer geöffneten spitzen Klammer „`<`" und wird durch eine geschlossene spitze Klammer „`>`" beendet. Beginn und Ende einer Auszeichnung werden in den meisten Fällen durch ein Tag-Paar definiert. In Zeile 9 ist zum Beispiel `<h1>...</h1>` zu sehen. Das zeigt den Inhalt in diesem Tag-Paar als eine Überschrift erster Ordnung. In Zeile 10 beginnt mit `<p>` ein Absatz, in dem zusätzlich eine Verschachtelung enthalten ist. Zusätzlich ist es möglich, in den Tags Attribute zu definieren, wie zum Beispiel in Zeile 12 das Attribut `alt`.

Neben den hier beschriebenen SGML-Elementen gibt es zahlreiche weitere, wie zum Beispiel

für den Header:

- base: Gibt entweder eine Basis-URI oder einen Basisframe an.
- link: Dient zur Verlinkung zu anderen Ressourcen. Wird am häufigsten zur Einbindung von Stylesheets benutzt.
- script: Verlinkt den Code in einer bestimmten Skriptsprache, primär JavaScript.

für den Body:

- a: Verlinkt eine URI-Adresse
- p: Leitet einen Textabsatz ein
- br: Erzwingt einen Zeilenumbruch
- strong: Setzt den Inhalt fett
- img: Bindet ein Bild eines beliebigen Formates ein

Vor- und Nachteile

Tabelle 11: Vor- und Nachteile des Dateiformates HTML ([54])

Vorteile	Nachteile
Hypertext-Möglichkeiten, z.B. Navigation im Text	Prozedere nicht einfach programmierbar
Links aus dem Text heraus, z.B. direkte Navigation zu anderen Ressourcen	Datenbankanwendungen nicht einfach integrierbar
Tabellen sind leicht zu erstellen	Virusanfälligkeit steigt mit der Verwendung von Scriptsprachen
Listen sind leicht erstellbar	Mehrere Versionen von HTML vorhanden
Dokument leicht druckbar	Für Styles andere Syntax
Bilder sind leicht integrierbar	Bilder-Platzierung ist umständlich
Technologie ist sehr weit verbreitet	Dynamische Erstellung mit weiteren Hilfsmitteln
HTML ist frei verfügbar	Abweichende Darstellung in unterschiedlichen Browsern
Von den meisten Systemen unterstützt	

3.3.3 Markdown

Geschichte / Grund der Entwicklung

Markdown ist eine vereinfachte Auszeichnungssprache, die von John Gruber und Aaron Swartz entworfen worden ist und im Dezember 2004 mit Version 1.0.1 veröffentlicht wurde.

Das Ziel von Markdown ist es, in der reinen Textform so einfach les- und schreibbar wie möglich zu sein, ohne dass die Syntax vor dem Betrachten umgewandelt werden muss. Daher werden für die Auszeichnungselemente vor allem Auszeichnungsarten verwendet, die in Plain-Texten und E-Mails üblich sind. So soll es möglich sein, die Grundform zu veröffentlichen, ohne dass Tags oder Formulierungsbefehle die Lesbarkeit stören, so wie es zum Beispiel bei HTML oder Latex der Fall ist [55].

Die Nutzung von Markdown findet überwiegend im technischen Umfeld statt. Populäre Beispiele sind GitHub, welches Markdown für die Readme-Dateien verwendet, Stack Overflow oder Blogging-Plattformen wie Ghost [55]. Des Weiteren verwendet die Presseabteilung der dpa für seine Veröffentlichungen seit 2010 Markdown [56]. Darüber hinaus wird Markdown in vielen weiteren Systemen verwendet, was ihre Beliebtheit stetig ansteigen lässt.

Aufbau

In Auflistung 11 [57] ist ein Minimal-Beispiel eines Markdown-Dokuments dargestellt.

```
1  Normaler Text wird so dargestellt wie eingegeben.
2
3  Eine Leerzeile erzeugt einen Absatz.
4  Zwei oder mehr Leerzeichen am Ende der Zeile
5  erzeugen einen Zeilenumbruch.
6
7  *Kursiv*, **Fett** und ***Fett kursiv*** bzw.
8  _Kursiv_, __Fett__ und ___Fett kursiv___
9
10 # Ueberschrift in Ebene 1
11 #### Ueberschrift in Ebene 4
12
13 > Dieses Zitat wird in ein HTML-Blockquote-Element integriert.
14
15 ![nur ein Beispiel](https://exsample.org/bilder/bild.jpg "Beispielbild")
```

Auflistung 11: Markdown-Datei `markdown.md`

Markdown besitzt keine besondere Einleitungssyntax. In einem Dokument kann sofort mit dem Schreiben eines Textes begonnen werden, was in Zeile 1 zu sehen ist. Mit einer Leerzeile wird ein Zeilenumbruch eingeleitet (Zeile 2).

Mit dem Zeichen * ... * wird der Text innerhalb der Sterne kursiv ausgegeben, mit zwei Sternen ** ... ** fett und mit drei Sternen fett und kursiv. Dies wird in den Zeilen 7 und 8 dargestellt. Überschriften werden mit Rauten eingeleitet. Eine Überschrift erster Ordnung definiert man mit einer Raute, eine Überschrift zweiter Ordnung mit zwei Rauten hintereinander usw. Dies wird in den Zeilen 10 und 11 dargestellt.

Einen Kommentar leitet man mit einer „>“ ein, welches in Zeile 13 demonstriert wird. Außerdem ist es möglich, Bilder zu integrieren. Hierzu wird mit einem Ausrufezeichen begonnen. Daran schließt in eckigen Klammern ein Alternativtext, gefolgt von runden Klammern, welche die URL zum anzuzeigenden Bild enthält.

Bietet Markdoown nicht genug Formulierungen an, können komplexere Ausdrücke aus dem XHTML-Blockelemente-Umfeld verwendet und Bereiche in gewöhnlichem XHTML formatiert werden. Darüber hinaus gibt es weitere Befehle, welche nicht weiter aufgeführt werden [57].

Vor- und Nachteile

Tabelle 12: Vor- und Nachteile des Dateiformates Markdown ([58])

Vorteile	Nachteile
Viele Systeme unterstützen Markdown	Bilder links- oder rechtsbündig zu platzieren, evtl. sogar Bildgrößen festzulegen ist im offiziellen Markdown nicht möglich
Markdown ist in vielen nativen Schreibprogrammen wie iAWriter zu finden	HTML-Klassen und Wrapper-Elemente fehlen zur Erstellung besserer Strukturen
Einfache Syntax	Zusätzliche eigene Syntax-Erweiterungen als Textfilter, welche den Standard aufweichen
Zahlreiche Erweiterungen	Gute Erweiterungen wurden nicht in einen „offiziellen“ Status aufgenommen

3.3.4 ReStructuredText

Geschichte / Grund der Entwicklung

ReStructuredText (RST) ist ebenso wie Markdown eine vereinfachte Auszeichnungssprache mit dem Ziel, in der reinen Textform besonders einfach les- und schreibbar zu sein. RST wurde von der Firma Zope entwickelt und wird seit 2002 von der Phyton Community verwendet. Weiterhin ist RST leicht in andere Formate umwandelbar. Mithilfe von RST können Textdokumente mit einfachen Strukturmerkmalen erzeugt werden. Im Hintergrund von RST arbeitet ein Parser, eine Komponente der Textverarbeitungsbibliothek Docutils, welche in der Programmiersprache Python entwickelten wurde. Mit Docutils ist es möglich, RST in verschiedene Formate zu überführen. Ab Version 0.6 wird die Umwandlung in ODT, XHTML, Latex und anderen ermöglicht. Für professionelle Schriftsätze ist auch die Konvertierung in PostScript und PDF durchführbar [59].

RST kommt unter anderem in der Formatierung der `docstrings`, der Python-typischen Kommentare zur Programmdokumentation, zum Einsatz. Des Weiteren bieten einige Wiki-Systeme neben ihrer eigenen Wiki-Syntax RST als Alternative an [59]. Um eine Datei mit RST-Inhalten zu erstellen, genügt es, eine neue Textdatei mit einem Editor zu öffnen. RST-Dateien werden mit der Endung `.rst` abgespeichert [60].

Aufbau

In Auflistung 12 [60] ist ein Minimal-Beispiel eines RST-Dokuments dargestellt.

```
1  Name eines Kapitels
2  ===================
3  Name eines Abschnitts
4  ---------------------
5  Name eines Unterabschnitts
6  ^^^^^^^^^^^^^^^^^^^^^^^^^^
7  Name eines Paragraphen
8  """""""""""""""""""""""
9  Kapitelname
10 ===========
11
12 Text.
13 .. rubric:: Bezeichnung
14
15 .. toctree::
16     :maxdepth: 2
17
18     kapitelname-1.rst
19     kapitelname-2.rst
20     kapitelname-3.rst
21
22 ..  Dies hier ist ein Kommentar.
23
24 - Ein Listen Item
25 - Zweites item
26   - Ein Unterabschnitt Item
27
28 1) Ein nummeriertes listen Item
29 2) Zweites Item
30    a) Ein Unterpunkt Item
31       i) Ein Unterunterpunkt Item
```

Auflistung 12: RST-Datei `ReStructuredText.rst`

RST benötigt keine spezielle Struktur. Der Autor beginnt mit dem Erstellen eines Dokuments und kann während des Schreibens die Formatierungen vornehmen. Will man zum Beispiel den Text mithilfe von Überschriften unterteilen, so stehen mehrere Überschriftgrade zur Verfügung. Es gibt die Möglichkeit, eine Überschrift mithilfe von Gleichheitszeichen zu

kennzeichnen, was in Zeile 1 und 2 demonstriert wird. Hiermit definiert man eine Überschrift ersten Grades. Eine Überschrift zweiten Grades wird durch Minuszeichen gekennzeichnet, welches in Zeile 3 und 4 zu sehen ist. Ein Paragraph wird mit Kommas eingeleitet. Zur Hervorhebung einer Überschrift muss somit nur die jeweilige Zeile durch eine zweite genauso lange Zeile mit entsprechenden Zeichen markiert werden. Anschließend muss eine leere Zeile folgen. Außerdem wird bei der Generierung einer PDF automatisch ein Inhaltsverzeichnis erstellt, in welchem die Überschriften aufgenommen werden. Will man dagegen erreichen, dass Unterpunkte ohne Nummerierung und Auflistung im Inhaltsverzeichnis erscheinen, so gibt es die Möglichkeit der Anwendung eines sogenannten „Direktive“ namens `.. rubric::`. Dies ist in der Zeile 13 und 14 dargestellt. Die weitere Aufgliederung eines Kapitels ist durch ein Aufspalten in mehrere Dateien möglich. Dabei werden die einzelnen Dateien mit dem Namen des gewünschten Kapitels benannt. In der Textdatei wird auf diese Dateien über ein Inhaltsverzeichnis verwiesen. Um ein Inhaltsverzeichnis (engl.: table of contents) zu erzeugen, wird die sogenannte „toctree“-Umgebung verwendet. Durch die Option `:maxdepth:` wird festgelegt, bis zu welcher Hierarchie-Stufe das Inhaltsverzeichnis aufgegliedert werden soll. Dieser Vorgang ist in den Zeilen 15 bis 20 dargestellt.

Des Weiteren sind Kommentare in das Dokument integrierbar. Dies wird durch zwei aufeinander folgende Punkte eingeleitet, was in Zeile 22 zu erkennen ist. Außerdem sind Listen sehr einfach zu erstellen, was in den Zeilen 24 bis 31 demonstriert wird. Darüber hinaus verfügt RST über zahlreiche weitere Befehle [60, 61].

Vor- und Nachteile

Tabelle 13: Vor- und Nachteile des Dateiformates RST ([60])

Vorteile	Nachteile
Viele Systeme unterstützen RST	Von Nutzern selten verwendet
RST in viele andere Formate umwandelbar	Konverter können Überschriften mit unterschiedlicher Schriftgröße und/oder Nummerierung darstellen
Einfache Syntax	Anwendung ist schwieriger als Markdown, daher zeitaufwendiger
Zahlreiche Auszeichnungsmöglichkeiten	

4 Gemeinsamkeiten und Unterschiede der einzelnen Formate

Tabelle 14: Gemeinsamkeiten und Unterschiede der einzelnen Formate

Format	**Kodierung**	**Verbreitung**	**Open Source**	**Plattform**	**Komplexität**	**Erweiterbarkeit**
BBCode	Markup, ähnlich HTML	- -	Ja	Windows, OSX, Linux	+	Nein
DITA	XML	- -	Ja	Windows, OSX, Linux	-	Nein
Doc	Binär	+	Nein	Windows	- - -	Nein
DocBook	XML, HTML, SGML	- -	Ja	Windows, OSX, Linux	-	Ja
EPUB	XML, HTML	+	Ja, mit Einschränkungen	Windows, OSX, Linux	- -	Nein
HTML	XML	+ + +	Ja	Windows, OSX, Linux	+	Ja
Latex	Markup	+	Ja	Windows, OSX, Linux	-	Ja
Markdown	Text, Markup	+	Ja	Windows, OSX, Linux	++	Ja
ODF	XML	++	Ja	Windows, OSX, Linux	- -	Nein
OOXML	XML	+++	Ja	Windows, OSX, (Linux)	- - -	Nein
PDF	Binär	+++	Ja	Windows, OSX, Linux	- -	Nein
RST	Text, Markup	- -	Ja	Windows, OSX, Linux	++	Nein
RTF	Text ANSI	+	Ja (z.T. Lizenzen)	Windows, OSX, Linux	+	Nein

5 Alleinstellungsmerkmale der einzelnen Formate

Tabelle 15: Alleinstellungsmerkmal der einzelnen Formate, Teil 1

Format	Alleinstellungsmerkmale
BBCode	- Vereinfachte HTML-Variante, unterstützt viele Tags - Verwendung gut für Forensoftware - Keine spezielle Struktur zur Erstellung nötig - Umwandlung in HTML möglich - Leicht zu implementieren - Unterstützt auch die Verwendung von Musik und Videos
DITA	- Erstellung von Büchern - Erstellung von Topics - Bibliotheken stehen zur Verfügung - Einfache Implementation
Doc	- Erstellung und Austausch von Bürodokumenten - Schwere Implementation - Kaum Bibliotheken vorhanden - Format veraltet, abgelöst durch OOXML - Unterstützt auch Verwendung von Musik (und Videos)
DocBook	- Erstellung von Büchern, Artikeln, Dokumentationen im technischen Umfeld - Gut zu implementieren - Open Source und Bibliotheken stehen zur Verfügung
EPUB	- Erstellung von E-Books mit dynamischer Anpassbarkeit an Bildschirmgröße - Gute Implementationsmöglichkeiten, da Open Source und Bibliotheken zur verfügbar stehen - Unterstützt auch Verwendung von Bildern, Musik und Videos
HTML	- Zur Strukturierung von Dokumenten, wie Texte, Bilder etc. - Einfache Erstellung von Dokumenten - Gut zu implementieren und zahlreiche Bibliotheken stehen zur Verfügung - Verwendung überwiegend im Web - Unterstützt auch Verwendung von Bildern, Musik und Videos
Latex	- Softwarepaket zur Erstellung von Dokumenten, Büchern, Artikeln etc. - Gute Implementierungsmöglichkeiten - Bibliotheken zur Erstellung vorhanden
Markdown	- Erstellung von strukturierten Texten, welche im Textform gut lesbar sind - Einfache Struktur, welche gut zu implementieren ist - Zahlreiche Bibliotheken stehen zur Verfügung - Verwendung überwiegend im technischen Umfeld

Tabelle 16: Alleinstellungsmerkmal der einzelnen Formate, Teil 2

Format	Alleinstellungsmerkmale
ODF	- Erstellung und Austausch von Bürodokumenten - Implementation kompliziert, da komplexe Struktur - Bibliotheken stehen zur Verfügung, was die Implementation erleichtert
OOXML	- Erstellung und Austausch von Bürodokumenten - Schwer zu implementieren, da sehr komplizierte Struktur und kaum Bibliotheken zur Verfügung stehen - Zusätzlich gibt es verschiedene Versionen des Formats
PDF	- Plattformunabhäniges Dateiformat für Erstellung / Austausch von Dokumenten - Layouttreue auf allen Systemen - Gut realisierbare Implementation, da zahlreiche Bibliotheken verfügbar - Vektororientiertes Format, damit auf viele Größenformate skalierbar
RST	- Erstellung von Dokumenten, welche leicht lesbar in der Textform sein sollen - Struktur der Texte soll nicht zu kompliziert sein - Leichte Implementierung, da zahlreiche Bibliotheken zur Verfügung stehen - Verwendung meistens im technischen Umfeld, wie in Wiki-Systemen
RTF	- Dient dem Austausch zwischen verschiedenen Textverarbeitungsprogrammen - Struktur ist nicht zu kompliziert - Aufgrund der Struktur ist die Implementation nicht aufwendig - Bibliotheken stehen zur Verfügung

6 Zusammenfassung

Mit diesem Paper wird eine Übersicht verfügbarer Textformate gegeben. In diesem Sinne werden die Formate in Block- und Markup-Formate eingeordnet. Dabei wurden die Textformate anhand verschiedener Aspekte betrachtet, wie zum Beispiel Hintergründe der Entwicklung oder der Aufbau. Betrachtet man die Vielzahl der verfügbaren Format, so erkennt man, dass viele auf XML basieren. Deshalb sind viele Formate ähnlich aufgebaut. Der Unterschied besteht jedoch in den verschiedenen Anwendungsszenarien, für die das Format entwickelt wurde. Einige Formate wie das Markup-Format OOXML legen das Augenmerk auf den stark vereinfachten Einstieg in die vielfältigen Funktionen für die Entwickler und Benutzer. Beispielsweise haben die Benutzer kaum oder kein Kontakt mit der komplizierten Struktur des Formates, da diese hinter einer Software verborgen bleibt. Außerdem verfügen die verschiedenen Formate über Anwendungen, mit der das Lesen der Texte leicht umsetzbar ist.

Die verschiedenen Formate müssen unter Umständen Altlast mit sich führen, um Kompatibilität aufrecht halten zu können. Das ist beispielsweise bei OOXML der Fall, um das Vorgänger-Format Doc im Block-Format unterstützen zu können. Die Recherche hat gezeigt, dass die Block-Formate zum größten Teil veraltet sind. Denn diese werden durch die Markup-Formate immer stärker verdrängt. Erkennbar ist das an der Anzahl der am häufigsten eingesetzten Formate.

In der Gesamtheit stehen zahlreiche Textformate zur Verfügung, die für viele unterschiedliche Aufgaben konzipiert wurden. Alle Formate haben dabei das Ziel, dem Benutzer einen besseren und vereinfachten Umgang mit Texten zu ermöglichen.

Literaturverzeichnis

[1] Microsoft: *Understanding Word's XML Markup [Word 2003 XML Reference] – Microsoft Office Word 2003 XML Software Development Kit*, Abgerufen am: 01.08.2017. https://msdn.microsoft.com/en-us/library/office/aa212889(v=office.11).aspx.

[2] Wikipedia: *Open Document Architecture*, Abgerufen am: 20.07.2017. https://en.wikipedia.org/wiki/Open_Document_Architecture.

[3] Microsoft: *[MS-DOC]: Word (.doc) Binary File Format*, Abgerufen am: 25.07.2017. https://msdn.microsoft.com/en-us/library/office/cc313153(v=office.12).aspx.

[4] Robbins, Jennifer Niederst: *Uebersicht zu HTML & XHTML*, Abgerufen am: 15.08.2017. https://www.data2type.de/xml-xslt-xslfo/html-und-xhtml/.

[5] W3C: *HTML5 Differences from HTML4*, Abgerufen am: 15.08.2017. https://www.w3.org/TR/html5-diff/.

[6] KG, Soft Xpansion GmbH & Co.: *PDF: Grundlagen eines Dateiformats*, Abgerufen am: 28.08.2017. http://soft-xpansion.eu/files/cc/PDF-Grundlagen.pdf.

[7] n2pdf: *PDF/A – Was ist das eigentlich? Und wofür benötigt man es?*, Abgerufen am: 28.08.2017. https://www.n2pdf.de/was-ist-n2pdf/pdfa-und-n2pdf/was-ist-pdfa.html.

[8] Connection, Adobe Developer: *PDF Reference and Adobe Extensions to the PDF Specification*, Abgerufen am: 28.08.2017. https://www.adobe.com/devnet/pdf/pdf_reference.html.

[9] Data2Type: *Uebersicht zu DITA*, Abgerufen am: 19.04.2017. https://www.data2type.de/xml-xslt-xslfo/dita/.

[10] Wikipedia: *Darwin Information Typing Architecture*, Abgerufen am: 19.08.2017. https://de.wikipedia.org/wiki/Darwin_Information_Typing_Architecture.

[11] OASIS: *OASIS Darwin Information Typing Architecture (DITA) TC*, Abgerufen am: 19.04.2017. https://www.oasis-open.org/committees/dita/faq.php.

[12] DocBook: *DocBook*, Abgerufen am: 19.08.2017. http://www.docbook.de/.

[13] Usegroup: *Einfuehrung in Docbook*, Abgerufen am: 19.08.2017. http://www.usegroup.de/software/xmltutorial/docbook.html.

[14] OASIS: *OASIS DocBook TC*, Abgerufen am: 19.08.2017. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=docbook.

[15] WikiBooks: *XML - Managing Data Exchange/DocBook*, Abgerufen am: 19.08.2017. https://en.wikibooks.org/wiki/XML_-_Managing_Data_Exchange/DocBook.

[16] Wikipedia: *DocBook*, Abgerufen am: 14.08.2017. https://en.wikipedia.org/wiki/DocBook.

[17] Eisentraut, Peter: *DocBook Textverarbeitung mit XML*, Abgerufen am: 19.08.2017. https://chemnitzer.linux-tage.de/2005/vortraege/shortpaper/docbook.pdf.

[18] Idpf: *EPUB*, Abgerufen am: 19.08.2017. http://idpf.org/epub.

[19] W3C: *New Roadmap for Future of Publishing is Underway as W3C and IDPF Officially Combine*, Abgerufen am: 19.08.2017. https://www.w3.org/2017/01/pressrelease-idpf-w3c-combination.html.en.

[20] W3C: *Epub 3.1*, Abgerufen am: 19.08.2017. https://www.w3.org/Submission/2017/SUBM-epub31-20170125/.

[21] Wikipedia: *Epub*, Abgerufen am: 19.08.2017. https://de.wikipedia.org/wiki/EPUB.

[22] Magazin, Linux: *Epub mit Open-Source-Tools erzeugen*, Abgerufen am: 19.08.2017. http://www.linux-magazin.de/Ausgaben/2011/09/Epub-Tools.

[23] EBook-Fiber, Joern: *Was ist ein eBook ? Vor- und Nachteile im Kurzueberblick*, Abgerufen am: 19.08.2017. https://ebook-fieber.de/ebooks/was-ist-ein-ebook-vor-und-nachteile-im-kurzuberblick/4675.

[24] E-teaching.org: *EPub*, Abgerufen am: 19.08.2017. https://www.e-teaching.org/technik/aufbereitung/text/e_pub.

[25] Lamport, Leslie: *My Writings*, Abgerufen am: 15.08.2017. http://lamport.azurewebsites.net/pubs/pubs.html#latex.

[26] Lamport, Leslie: *LATEX 2e for authors*, Abgerufen am: 15.08.2017. http://mirror.physik-pool.tu-berlin.de/tex-archive/macros/latex/doc/usrguide.pdf.

[27] e.V., Dante Professional Typesetting Deutschsprachige Anwendervereinigung TeX: *Was ist LaTeX?*, Abgerufen am: 15.08.2017. https://www.dante.de/tex/WasIstLaTeX.html.

[28] Voss, Herbert: *Einfuehrung in Latex*, Abgerufen am: 15.08.2017. https://books.google.de/books?id=z793CwAAQBAJ&pg=PA880&lpg=PA880&dq=latex+spezifikation&source=bl&ots=w529iHfRBd&sig=WeoUnNB3HBmTlSIgEgE5b3G_yRQ&hl=de&sa=X&ved=0ahUKEwio88X17abTAhXOJlAKHSy0BykQ6AEIUDAH#v=onepage&q=latex%20spezifikation&f=false.

[29] Wikipedia: *LaTeX*, Abgerufen am: 15.08.2017. https://de.wikipedia.org/wiki/LaTeX.

[30] mschemie.wordpress.com: *Vor- und Nachteile von LaTeX*, Abgerufen am: 15.07.2017. https://mschemie.wordpress.com/2009/11/26/vor-und-nachteile-von-latex/.

[31] Wiki, OpenOffice.org: *OpenDocument*, Abgerufen am: 13.07.2017. http://www.ooowiki.de/OpenDocument.html.

[32] Zendel, Oliver: *Das OASIS Open Document Format*, Abgerufen am: 13.07.2017. http://www.pro-linux.de/artikel/2/644/3,seite-3.html.

[33] bmi.bund: *IT-Rat der Bunderregierung eroeffnet den Einsatz offener Dokumentenformate (ODf)*, Abgerufen am: 13.07.2017. https://www.bmi.bund.de/cln_156/SharedDocs/Pressemitteilungen/DE/2008/12/odf.htmll.

[34] Microsoft: *Unterschiede zwischen dem OpenDocument-Textformat (ODT) und dem Word-Format (DOCX)*, Abgerufen am: 13.07.2017. https://support.office.com/de-de/article/Unterschiede-zwischen-dem-OpenDocument-Textformat-ODT-und-dem-Word-Format-DOCX-d9d51a92-56d1-4794-8b68-5efb57aebfdc.

[35] Wikipedia: *OpenDocument*, Abgerufen am: 13.07.2017. https://de.wikipedia.org/wiki/OpenDocument#Deutschland.

[36] Rice, Frank: *Einfuehrung in die Microsoft Office (2007) Open XML-Dateiformate*, Abgerufen am: 14.07.2017. https://msdn.microsoft.com/de-de/library/aa338205(v=office.12).aspx.

[37] Ecma: *What is Ecma International*, Abgerufen am: 14.07.2017. https://www.ecma-international.org/memento/index.html.

[38] Microsoft: *Ecma International Standardization of OpenXML File Formats Frequently Asked Questions*, Abgerufen am: 14.07.2017. http://www.microsoft.com/office/preview/itpro/ecmafaq.mspx.

[39] Netzpolitik: *Oh-oh-XML – Digitale Zeitbombe in deutschen Amtsstuben*, Abgerufen am: 14.08.2017. https://netzpolitik.org/2014/ooxml-digitale-zeitbombe-in-deutschen-amtsstuben/.

[40] data2type: *Grundaufbau WordML*, Abgerufen am: 14.08.2017. https://www.data2type.de/xml-xslt-xslfo/wordml/wordml-einfuehrung/grundaufbau0/.

[41] Microsoft: *Das Open XML SDK 2.0*, Abgerufen am: 14.08.2017. https://www.microsoft.com/germany/msdn/webcasts/serien/MSDNWCS-0901-02.aspx.

[42] PC-Magazin: *Office-Dateiformate im Test*, Abgerufen am: 14.07.2017. http://www.pc-magazin.de/ratgeber/office-dateiformate-im-test-131588.html.

[43] OsterBlog: *Die Probleme mit Office Open XML (OOXML)*, Abgerufen am: 14.08.2017. http://web.oesterchat.com/2007/09/05/probleme-mit-ooxml/.

[44] Golem: *Kopf einziehen und ueber Verschwoerung tuscheln*, Abgerufen am: 14.08.2017. https://www.golem.de/news/limux-kopf-einziehen-und-ueber-verschwoerung-tuscheln-1412-110908.html.

[45] Microsoft: *Word 2007: Rich Text Format (RTF) Specification, version 1.9.1*, Abgerufen am: 14.08.2017. https://www.microsoft.com/en-us/download/details.aspx?id=10725.

[46] Rouse, Margaret: *Rich Text Format (RTF)*, Abgerufen am: 14.08.2017. http://searchexchange.techtarget.com/definition/Rich-Text-Format.

[47] SEC, PAC: *OpenOffice / OpenDocument and MS Office 2007 / Open XML security*, Abgerufen am: 14.08.2017. https://pacsec.jp/psj06/psj06lagadec-e.pdf.

[48] Wikipedia: *Rich Text Format*, Abgerufen am: 14.08.2017. https://de.wikipedia.org/wiki/Rich_Text_Format.

[49] Limited, Solid Documents: *Warum soll ich das RTF-Format (Rich Text Format) verwenden?*, Abgerufen am: 14.08.2017. http://www.soliddocuments.com/de/pdf/_the_rtf_file_format/65/1.

[50] germany.com vbulletin: *BBCode*, Abgerufen am: 19.08.2017. http://forum.vbulletin-germany.com/misc.php?do=bbcode.

[51] Websolution, My: *BBCode*, Abgerufen am: 19.08.2017. http://www.mywebsolution.de/tutorials/4/show_BB-Code.html.

[52] TEIA: *Geschichte von HTML*, Abgerufen am: 15.08.2017. https://www.teialehrbuch.de/Kostenlose-Kurse/HTML/3260-Geschichte-von-HTML.html.

[53] SELFHTML: *HTML*, Abgerufen am: 15.08.2017. https://wiki.selfhtml.org/wiki/HTML.

[54] MADEASY: *Vor - und Nachteile der HTML Sprache*, Abgerufen am: 15.08.2017. http://www.madeasy.de/2/html.htm.

[55] MARKDOWN.DE: *Markdown Übersicht*, Abgerufen am: 18.08.2017. http://markdown.de/.

[56] GKAMP: *dpa-Notizblock und die seltsamen #-Zeichen*, Abgerufen am: 18.08.2017. https://web-beta.archive.org/web/20150402181810/http://www.dpa-newslab.com/2010/04/22/dpa-notizbuch-markdown/.

[57] PRITCHARD, ADAM: *Markdown Cheatsheet*, Abgerufen am: 18.08.2017. https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet.

[58] AAKEN, GERRIT VAN: *Die Großartigkeit von (und das Problem mit) Markdown*, Abgerufen am: 17.08.2017. https://praegnanz.de/weblog/markdown.

[59] RESTRUCTUREDTEXT: *reStructuredText*, Abgerufen am: 18.08.2017. http://docutils.sourceforge.net/rst.html.

[60] WISSEN.DE HTTP://WWW.GRUND: *ReStructuredText-Elemente*, Abgerufen am: 18.08.2017. http://www.grund-wissen.de/linux/sphinx/rst-tutorial.html.

[61] DOCUTILS.SOURCEFORGE.NET: *reStructuredText Directives*, Abgerufen am: 18.08.2017. http://docutils.sourceforge.net/docs/ref/rst/directives.html.